FORMAL MODELLING AND ANALYSIS OF DENIAL OF SERVICES ATTACKS IN WIRELESS SENSOR NETWORKS

KASHIF SAGHAR MALIK

A thesis submitted in partial fulfilment
of the requirements of the
University of Northumbria at Newcastle
for the degree of
Doctor of Philosophy

Research undertaken in the School of Computing, Engineering and Information Sciences

November 2010

ABSTRACT

Wireless Sensor Networks (WSNs) have attracted considerable research attention in recent years because of the perceived potential benefits offered by self-organising, multi-hop networks consisting of low-cost and small wireless devices for monitoring or control applications in difficult environments. WSN may be deployed in hostile or inaccessible environments and are often unattended. These conditions present many challenges in ensuring that WSNs work effectively and survive long enough to fulfil their functionalities. Securing a WSN against any malicious attack is a particular challenge. Due to the limited resources of nodes, traditional routing protocols are not appropriate in WSNs and innovative methods are used to route data from source nodes to sink nodes (base stations). To evaluate the routing protocols against DoS attacks, an innovative design method of combining formal modelling and computer simulations has been proposed. This research has shown that by using formal modelling hidden bugs (e.g. vulnerability to attacks) in routing protocols can be detected automatically. In addition, through a rigorous testing, a new routing protocol, RAEED (Robust formally Analysed protocol for wirEless sEnsor networks Deployment), was developed which is able to operate effectively in the presence of hello flood, rushing, wormhole, black hole, gray hole, sink hole, INA and jamming attacks. It has been proved formally and using computer simulation that the RAEED can pacify these DoS attacks. A second contribution of this thesis relates to the development of a framework to check the vulnerability of different routing protocols against Denial of Service (DoS) attacks. This has allowed us to evaluate formally some existing and known routing protocols against various DoS attacks iand these include TinyOS Beaconing, Authentic TinyOS using uTesla, Rumour Routing, LEACH, Direct Diffusion, INSENS, ARRIVE and ARAN protocols. This has resulted in the development of an innovative and simple defence technique with no additional hardware cost for deployment against wormhole and INA attacks. In the thesis, the detection of weaknesses in INSENS, Arrive and ARAN protocols was also addressed formally. Finally, an efficient design methodology using a combination of formal modelling and simulation is propose to evaluate the performances of routing protocols against DoS attacks.

Contents

1 Introduction **1**

1.1 Wireless Sensor Networks (WSNs) 1

1.1.1 Differences Between WSNs and Other Networks 3

1.1.2 Applications of WSN 3

1.2 Routing and Security in WSN 4

1.3 The Dissertation 5

1.3.1 Motivation and Overview 5

1.3.2 Problem Statement 6

1.3.3 Main Contributions to Research 6

1.3.4 Structure of the Dissertation 7

2 WSN, Security and Formal Modelling **9**

2.1 Introduction 9

2.2 Routing in WSN 10

2.3 Attacks and Security Issues in WSNs 12

2.4 Denial of Service Attacks 13

2.4.1 Spoofing Attack 14

2.4.2 False Injection Attack 14

2.4.3 Hello Flood Attack 15

2.4.4 Wormhole Attack and Invisible Node Attack (INA) 15

2.4.5 Sinkhole Attack 17

2.4.6 Sybil Attack 17

2.4.7 Rushing Flood Attack 18

2.4.8 Jamming Attack 18

2.4.9 Black hole Attack and Selective Forwarding Attack 19

2.5 Classification of Secure/Robust Routing Protocols 20

2.5.1 Protocols using Multiple-paths 21

2.5.2 Probabilistic Path Selection Protocols 22

2.5.3 Protocols that Overhear Neighbor Communication 22

2.5.4 Protocols using Specialized Hardware 22

2.5.5 Topology Mapping Protocols . . . 23
2.5.6 Protocols using Cryptographic Techniques . . . 23
2.6 Formal Modelling and WSN . . . 23
2.6.1 Formal Modelling in WSNs . . . 24
2.6.2 Formal Modelling of Routing Protocols . . . 24
2.6.3 Formal Modelling and Security in WSN . . . 25
2.7 Chapter Summary . . . 26

3 Proposed Formal Specifications of WSNs and Attacks 27
3.1 Introduction and Motivation . . . 27
3.2 Motivation . . . 27
3.3 Proposed Formal Specifications for WSN . . . 28
3.3.1 Basic Definitions . . . 28
3.3.2 Basic Operations . . . 29
3.3.2.1 Receive Operation . . . 29
3.3.2.2 Transmit Operation . . . 30
3.3.2.3 In Range Operation . . . 31
3.3.2.4 Connected Operation . . . 31
3.3.2.5 Neighbour Definitions . . . 31
3.3.2.6 Cryptography . . . 32
3.3.2.7 Eavesdrop Key . . . 33
3.3.2.8 Node Capture . . . 33
3.3.2.9 Encryption Fail . . . 34
3.4 Proposed Formal Specifications for Denial of Service Attacks . . . 34
3.4.1 Wormhole Attack . . . 34
3.4.2 Invisible Node Attack (INA) . . . 35
3.4.3 Black hole Attack . . . 36
3.4.4 Spoofing Attack . . . 37
3.4.5 False-Injection Attack . . . 38
3.4.6 Sybil Attack . . . 38
3.4.7 Node Replication Attack . . . 39
3.4.8 Hello-Flood Attack . . . 39
3.4.9 Jamming Attack . . . 40
3.4.10 Sinkhole Attack . . . 40
3.5 Chapter Summary . . . 41

4 Formal Analysis of Routing Protocols 42
4.1 Introduction . . . 42
4.2 Methods Adopted in the Research . . . 43

4.3 Formal Framework . . . 44
4.3.1 The Framework . . . 44
4.3.2 Modelling Topologies . . . 45
4.3.3 Assumptions for the Formal Framework . . . 47
4.3.4 Modelling a Channel . . . 47
4.4 The Flooding Protocol . . . 48
4.4.1 Semi Formal Notation . . . 48
4.4.2 Formal Model . . . 48
4.4.3 Verification . . . 51
4.4.4 The Trace . . . 52
4.5 The TinyOS Beaconing Protocol . . . 52
4.5.1 Protocol Description . . . 52
4.5.2 Formal Model . . . 53
4.5.3 Verification . . . 55
4.5.4 Black hole Attack . . . 55
4.5.5 Sinkhole Attack . . . 55
4.5.6 Wormhole/INA Attack . . . 56
4.5.7 Hello Flood Attack . . . 57
4.5.8 Spoofing Attack . . . 57
4.5.9 Other DoS Attacks . . . 57
4.6 The Authentic TinyOS Protocol using uTESLA . . . 58
4.6.1 Protocol Description . . . 58
4.6.2 Formal Model . . . 58
4.6.3 Performance of Authentic TinyOS against DoS attacks . . . 59
4.7 Other Unsecured Routing Protocols . . . 59
4.7.1 The Direct Diffusion Protocol . . . 60
4.7.2 The Rumour Routing Protocol . . . 61
4.8 The Enhanced INSENS Protocol . . . 62
4.8.1 Protocol Description . . . 62
4.8.2 The Formal Model . . . 63
4.8.3 Invisible Node Attack (INA) . . . 65
4.8.3.1 Formal Analysis . . . 65
4.8.3.2 Simulation Results . . . 65
4.8.4 Wormhole Attack . . . 67
4.8.5 Black hole Attack . . . 68
4.8.5.1 Formal Analysis . . . 68
4.8.5.2 Simulation Results . . . 68
4.9 Arrive Routing Protocol . . . 70

4.9.1 Protocol Description . . . 70
4.9.2 Formal Model . . . 71
4.9.2.1 Model . . . 71
4.9.2.2 Verification . . . 73
4.9.3 Black hole Attack . . . 74
4.9.3.1 Black hole Attack with a Single Path . . . 74
4.9.3.2 Black hole Attack with Multiple Paths . . . 75
4.9.4 INA . . . 76
4.9.5 Wormhole Attack . . . 76
4.9.6 Other DoS Attacks . . . 77
4.10 ARAN Routing Protocol . . . 78
4.10.1 Protocol Description . . . 78
4.10.2 Formal Model . . . 79
4.10.2.1 Model . . . 79
4.10.2.2 Verification . . . 81
4.10.3 Worm Hole Attack/INA . . . 82
4.10.4 Black Hole Attack . . . 83
4.11 Summary . . . 83

5 A Proposed New Protocol RAEED: Design and Evaluation 84

5.1 Introduction . . . 84
5.2 Evaluation of Secure Routing Protocols . . . 85
5.3 The New Protocol: An Overview . . . 86
5.4 Message Format . . . 89
5.5 Key Setup Phase (KSP) . . . 90
5.5.1 The Design . . . 90
5.5.2 Formal Verification of Bidirectional Property . . . 93
5.5.2.1 Formal Model . . . 93
5.5.2.2 Verification . . . 94
5.5.3 Simulation Results to Evaluate Efficiency . . . 94
5.5.4 Effect of Noise and Collision on the KSP . . . 96
5.5.4.1 Formal Model . . . 96
5.5.4.2 Simulation Results . . . 100
5.5.5 Confirmation that Security Properties Hold . . . 101
5.5.5.1 Formal Model . . . 102
5.5.5.2 Verification . . . 103
5.5.6 Simulation Results: Effect of Different Factors on KSP . . . 103
5.5.6.1 Effect of Time to Send ASK (Delay) on KSP . . . 103

5.5.6.2 Effect of Time to Send ASSIGN (Delay) on KSP . . . 105
5.5.6.3 Effect of Redundancy in ASK . . . 106
5.5.6.4 Effect of Redundancy in ASK with 2 Assign Attempts . . . 107
5.6 Route Setup Phase (RSP) . . . 109
5.6.1 Available Techniques for Route Setup . . . 110
5.6.2 The Proposed Design . . . 111
5.6.3 Neighbour Propagation Phase (NPP) . . . 111
5.6.4 Loud Test Phase (LTP) . . . 113
5.6.5 Level Propagation Phase (LPP) . . . 114
5.6.6 Node Synchronization Phase (NSP) . . . 116
5.6.7 Message Sequence Diagram . . . 117
5.6.8 An Example to Explain Level Propagation Phase . . . 118
5.6.9 Formal Verification of SPP and LPP . . . 120
5.6.9.1 Model . . . 120
5.6.9.2 Verification . . . 121
5.6.10 Simulation Results without Synchronization Propagation . . . 122
5.6.10.1 Effect of Density . . . 122
5.6.10.2 Effect of Level Propagation Delay . . . 124
5.6.10.3 Effect of Utilization Time . . . 125
5.6.11 Simulation Results after Synchronization Propagation . . . 126
5.6.11.1 Effect of Synchronous Propagation Delay . . . 127
5.6.11.2 Effect of Utilization Time . . . 128
5.6.11.3 Effect of Level Propagation Delay . . . 130
5.6.11.4 Simulation Results Summary . . . 132
5.6.12 Simulation Results with Multiple BSs . . . 132
5.6.12.1 Effect of Density . . . 133
5.6.12.2 Effect of Density with Slot Time for each BS . . . 136
5.6.12.3 Effect of Scalability and Density . . . 140
5.7 Data Forwarding Phase (DFP) . . . 143
5.7.1 DFP Design:Lost Indication Scheme . . . 144
5.7.2 Evaluation of Lost Indication Scheme . . . 145
5.7.2.1 Formal Verification of Lost Indication Scheme . . . 145
5.7.2.2 Simulation Results . . . 148
5.7.3 Effect of Noise on DFP . . . 148
5.7.3.1 Formal Verification . . . 148
5.7.3.2 Simulation Results . . . 148
5.8 Chapter Summary . . . 150

6 Evaluation of RAEED Against DoS Attacks 152
6.1 Introduction . . . 152
6.2 Prevention Against the Hello Flood Attack . . . 152
6.2.1 Formal Verification . . . 153
6.2.2 Computer Simulation . . . 154
6.2.3 Practical Implementation . . . 154
6.3 Prevention Against the INA and Wormhole Attack . . . 155
6.3.1 Traditional INA and Wormhole Attack . . . 155
6.3.2 Formal Verification . . . 156
6.3.2.1 Model . . . 156
6.3.2.2 Verification . . . 160
6.3.2.3 A Successful Wormhole Attack . . . 161
6.3.3 Practical Implementation . . . 161
6.3.4 Intelligent Wormhole/INA (Encryption Failure) . . . 162
6.3.4.1 Model . . . 162
6.3.4.2 Verification . . . 163
6.3.5 Intelligent Wormhole Attacker with Signal Detection . . . 164
6.4 Prevention Against the Sinkhole Attack . . . 165
6.4.1 Formal Modelling . . . 165
6.4.1.1 Model . . . 165
6.4.1.2 Verification . . . 166
6.4.2 Computer Simulation . . . 167
6.5 Prevention Against the Tunnel Attack . . . 167
6.5.1 Formal Modelling . . . 168
6.5.2 Computer Simulation . . . 169
6.5.3 Tunnel Attack in Combination with Framing Attack . . . 169
6.6 Prevention Against the Rushing Attack . . . 170
6.7 Prevention Against the Black hole Attack . . . 171
6.7.1 Formal Verification . . . 173
6.7.2 Simulation Results . . . 173
6.7.2.1 A 1000 Node Network with Black holes . . . 173
6.7.2.2 A 200 Node Network with Black holes . . . 175
6.7.3 Intelligent Black hole Attack Prevention . . . 177
6.8 Prevention Against the Gray hole Attack . . . 180
6.8.1 Formal Verification . . . 180
6.8.2 Simulation Results . . . 181
6.9 Prevention Against the Jamming Attack . . . 181
6.9.1 Formal Verification . . . 181

6.9.2 Computer Simulation . . . 182
6.10 Chapter Summary . . . 183

7 Conclusions and Future Work **184**
7.1 Conclusions . . . 184
7.2 Summary of the Contribution to Research . . . 188
7.2.1 Main Contribution to Research . . . 188
7.2.2 Other Contributions to Research . . . 189
7.3 Future Work . . . 189

A Assumptions, Methods and Database **192**
A.1 Introduction . . . 192
A.2 Assumptions . . . 192
A.2.1 Network Assumptions . . . 192
A.2.1.1 Node Placement and Topology . . . 192
A.2.1.2 Network Size . . . 193
A.2.1.3 Node Density . . . 193
A.2.1.4 Limited Resources . . . 193
A.2.1.5 Powerful Base Station . . . 193
A.2.2 Radio Links Assumptions . . . 193
A.2.2.1 Circular Links . . . 193
A.2.2.2 Bidirectional/Unidirectional Links . . . 194
A.2.2.3 Adjustable Radio Transmission Power . . . 194
A.2.2.4 Ideal Link Layer . . . 194
A.2.3 Cryptography Assumptions . . . 194
A.2.4 Attacker Assumptions . . . 194
A.3 Formal Modeling Tool: Uppaal . . . 195
A.4 Computer Simulation . . . 197
A.4.1 Simulator Selected: TOSSIM . . . 197
A.4.2 Aims of Simulation in the Thesis . . . 198
A.4.3 Noise in Computer Simulation . . . 199
A.5 Practical Implementation . . . 200
A.6 DFP: The Handshake Scheme . . . 200
A.6.1 Design of the Handshake Scheme . . . 200
A.6.2 Comparison of the Handshake Scheme with Flooding . . . 201
A.6.2.1 Visual Inspection . . . 202
A.6.2.2 Formal Modelling . . . 203
A.7 Database Required in the New Protocol . . . 205
A.7.1 Neighbour Table (NT) . . . 206

A.7.2 BS Information Table (BT) . . . 207
A.7.3 Event Table (ET) . . . 208
A.7.4 Data storage Table (DT) . . . 208
A.7.5 Forward Table (FT) . . . 209

B Verification of Claims **210**

B.1 Claims on Models of Different Routing Protocols . . . 210
B.1.1 Arrive Routing Protocol Claims . . . 210
B.1.2 ARAN Routing Protocol Claims . . . 211
B.2 Claims on the Models of RAEED Routing Protocol . . . 212
B.2.1 Bidirectional Property Claim in RAEED . . . 212
B.2.2 Effect of Noise and Collision on the KSP . . . 213
B.2.3 Confirmation that Security Properties Hold . . . 213
B.2.4 Verification of SPP and LPP (RSP) . . . 214
B.2.5 Verification of Lost Indication Scheme (DFP) . . . 216
B.3 Claims involving the Evaluation of RAEED Against the Attacks . . . 217
B.3.1 Prevention Against the INA and Wormhole Attack . . . 217
B.3.2 Prevention Against the Intelligent Wormhole Attack . . . 219
B.3.3 Prevention Against the Sinkhole Attack . . . 219

List of Figures

1.1 Wireless Sensor Network 2

2.1 Attacks in WSN 13

4.1 Approach adopted in the research 43
4.2 Examples of Connectivity Matrix 46
4.3 Connectivity Matrix Implementation 47
4.4 UPPAAL model for Flooding protocol 49
4.5 Confirmation by Uppaal that a topology satisfies the property 50
4.6 Trace generated by Uppaal in case the property fails 51
4.7 Node model for the TinyOS beaconing Protocol 53
4.8 Uppaal model for the TinyOS beaconing Protocol 54
4.9 Traces for a topology in which the formal model detected successful attacks in the TinyOS protocol 56
4.10 Trace for a topology for which formal model detected successful attacks in Authentic TinyOS (uTESLA) protocol 60
4.11 Traces for a topology in which the formal model detected successful sinkhole attack in the Directed Diffusion protocol 61
4.12 Uppaal models for Enhanced INSENS protocol 64
4.13 Node model for Phase III & IV (Enhanced INSENS) 68
4.14 Node model for the Arrive Protocol 71
4.15 UPPAAL models for the Arrive Protocol 73
4.16 Trace of Uppaal showing that black hole is possible in Arrive 74
4.17 Trace of Uppaal showing that wormhole is possible in Arrive 76
4.18 Trace of Uppaal showing other DoS attacks are possible in Arrive 77
4.19 Uppaal models for the ARAN Protocol 79
4.20 Uppaal models for the ARAN Protocol 80
4.21 Node model of the ARAN Protocol 82
4.22 Trace of UPPAAL showing other DoS attacks are possible in ARAN 83

5.1 RAEED Protocol Phases 88

5.2 Two cases in initiating the Bidirectional Verification Phase 91
5.3 Uppaal model for improved Key Setup Phase in RAEED 93
5.4 The effect of density on total messages exchanged in KSP 95
5.5 Uppaal model to check effect of message loss on INSENS with separate channel model 96
5.6 Uppaal model to check effect of message loss on INSENS without separate channel model 98
5.7 UPPAAL model to check effect of message loss on RAEED 99
5.8 The percentage of messages lost in the KSP in 100 node grid network 100
5.9 Uppaal model to check security of INSENS and RAEED is same 102
5.10 The effect of time to send ASK (delay) in the KSP 104
5.11 The effect of of time to send ASSIGN (delay) on Collision in the KSP 105
5.12 The effect of sending redundant ASK messages in KSP 106
5.13 The effect of sending redundant ASK messages in KSP 108
5.14 The effect of density on percentage of Neighbour message lost in 100 node network 112
5.15 Message Sequence Diagram for RSP 117
5.16 A 9 node network to explain level propagation 119
5.17 Sink model to verify the SPP and LPP of RAEED 120
5.18 Node model to the SPP and LPP of RAEED 121
5.19 The effect of density on level propagation in RSP 123
5.20 The effect of Level propagation delay on RSP 125
5.21 The effect of utilization time on RSP 126
5.22 The effect of delay in sending the SYNCHRONOUS message on RSP 128
5.23 The effect of different time spans used to send LEVEL beacons in RSP 129
5.24 The effect of Level propagation delay on RSP 131
5.25 The effect of density on RSP using multiple BSs 134
5.26 The error percentage in NT entries using multiple BS 136
5.27 The effect of density on RSP using multiple BSs with independent slot time . . . 137
5.28 The effect of density on RSP error percentage in NT entries using multiple BS with independent slot time 139
5.29 The effect of scalability on RSP using multiple BSs 140
5.30 The effect of scalability on RSP error percentage in NT entries using multiple BS 142
5.31 Node model used in verification of DFP 146
5.32 Sink model used in verification of DFP 147
5.33 The effect of noise on the DFP 149

6.1 Models used in the hello flood attack solution 153
6.2 Event generator model used in the wormhole solution 157

6.3 Attacker models . . . 157
6.4 Node model to check the wormhole attack . . . 158
6.5 Topologies used in the wormhole attack . . . 160
6.6 One of the topologies used in a hardware implementation of INA . . . 161
6.7 Wormhole Attacker Model . . . 162
6.8 Node model to check the wormhole attack . . . 163
6.9 Sink model to check the sinkhole attack . . . 166
6.10 Node model to check the sinkhole attack . . . 167
6.11 Attacker model to check the sinkhole attack . . . 168
6.12 Sink model to check the tunnel attack . . . 169
6.13 Node model to check the tunnel attack . . . 170
6.14 Attacker model to check the tunnel attack . . . 171
6.15 Modified node model to check the tunnel attack . . . 172
6.16 Black hole Attacker model . . . 173
6.17 Percentage of nodes blocked due to black hole in 1000 node network . . . 174
6.18 Case 1:The effect of the black hole attack on INSENS and RAEED in a 200 node network with a density of 8 and asymmetric attacker position . . . 175
6.19 Case 2: The effect of the black hole attack on INSENS and RAEED in a 200 node network with a density of 8 and symmetric attacker position . . . 177
6.20 Intelligent black hole attacker model . . . 178
6.21 Gray hole attacker model . . . 180
6.22 Attacker model to test the Jamming Attack . . . 181
6.23 Node model to test the Jamming Attack . . . 182

A.1 An example of Transmitter/Receiver system to explain Uppaal modelling . . . 195
A.2 Percentage of message lost due to noise . . . 199
A.3 Message sequence diagram to describe DFP (Handshake scheme) . . . 201
A.4 Sink model used in UPPAAL to compare results with flooding . . . 204
A.5 Node model used in UPPAAL to compare results with flooding . . . 204
A.6 Database Interface Diagram . . . 206

List of Tables

4.1 The number of symmetric and asymmetric topologies for a given number of nodes 46

5.1 Table comparing number of messages in KSP of RAEED,LEAP and INSENS . . 92
5.2 Noise samples categorized by samples of 10k used to test noise 101
5.3 Parameters used in different experiments on Key Setup Phase parameters 104
5.4 Assigned levels and time spans when `TIMELEVEL` is 100 msec 119
5.5 Levels and time spans when `TIMELEVEL` is 50 msec 119
5.6 Parameters used to perform simulations without synchronization propagation in RSP 122
5.7 Parameters used in experiments performed using the Synchronization Propagation in RSP 127
5.8 Parameters used to check the effect of Level propagation delay on RSP 133
5.9 Delay time introduced in multiple BSs for 100 node network 138
5.10 Delay time introduced in multiple BSs for 1000 node network 141

A.1 Parameters used to check the effect of Level propagation delay on RSP 203

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and thanks to my supervisors Dr William Henderson, Dr. David Kendall and Professor Ahmed Bouridane for their generous provision of their time, advice, guidance and patience throughout the whole project and dissertation process. Finally, I would like to express my gratitude to my parents, my wife, little son and other family members whom prayers and wishes were always with me during the whole course of my studies.

DECLARATION

I declare that the work contained in this thesis has not been submitted for any other award and that it is all my own work.

Name: Kashif Saghar Malik

Signature:

Date: 04 November 2010

PUBLISHED WORK

The work presented in the thesis is entirely my own, except where explicitly acknowledged. The papers, in chronological order, are:

- K. Saghar, W. Henderson, and D. Kendall. Formal modelling and analysis of routing protocol security in wireless sensor networks. *In PGNET '09*, June 2009.
- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Formal modelling of a robust wireless sensor network routing protocol. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2010)*, June 2010.
- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Applying Formal modelling to detect DoS attacks in wireless medium. In *IEEE, IET International Symposium on COMMUNICATION SYSTEMS, NETWORKS AND DIG- ITAL SIGNAL PROCESSING NASA/ESA(CSNDSP 2010)*, July 2010.

Submitted/Future Papers

- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Formal specifications of denial of service attacks in wireless sensor networks. Submitted at Pervasive and Mobile Computing Journal, 2010.
- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Automatic detection of black hole attack in wireless network routing protocols. Submitted at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011.
- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. An innovative solution for the INA and wormhole attack in wireless sensor networks (WSNs). Submitted at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011.
- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Vulnerability of INSENS to denial of service attacks. 2010.
- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. RAEED - A formally evaluated routing protocol for WSN against DoS attacks. 2010.

- K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Formal Modelling of Wireless Sensor Networks. 2010.

Chapter 1

Introduction

1.1 Wireless Sensor Networks (WSNs)

Advances in digital and radio frequency microelectronics have enabled the development of low cost, low power, small sized, multi-functional embedded devices called nodes (also referred to as motes to emphasise their small size) which can communicate wirelessly. These individual nodes, in spite of their limitations (restricted in CPU speed, RF power, memory capacity and bandwidth) and simple radio communication, may be combined together to physically sense and report information relating to their environment (temperature, vibration, humidity, light, radiation, etc). This combination (usually a large number) of the distributed nodes is called a Wireless Sensor Network (WSN). The nodes in a WSN, by communicating with one another in a multi-hop network with a base station (BS), can collect data over a large geographical area. WSN can be deployed randomly in inaccessible areas and the positions of the nodes need not be predetermined.

WSN nodes are usually battery powered and may be required to operate for long periods. When batteries get exhausted, a node becomes out of service and network performance is degraded. Wireless transmission and reception makes significant demands on available energy in addition to the need to process the data, sense the environment etc. Thus, energy efficiency is an important consideration in a protocol design for multi-hop networks like WSNs. This had strong influence on the design of protocols.

The nodes that are used to detect environmental *events* and send these as message to the other nodes are called *source* nodes; while the nodes that are dedicated as gathering points are called the *sink* nodes. The sink nodes absorb message packets and do not retransmit data messages. Sometimes special nodes called *base stations* are also deployed, which contain higher resources (laptop class) than a normal node; these normally control the WSN and gather all the environmental data. A base station (BS) is always a sink but a sink might not be a BS. The remaining nodes of a network act as routing/intermediate nodes and are used to forward data from the sources to the sinks. The number of sinks or sources required in a network depends

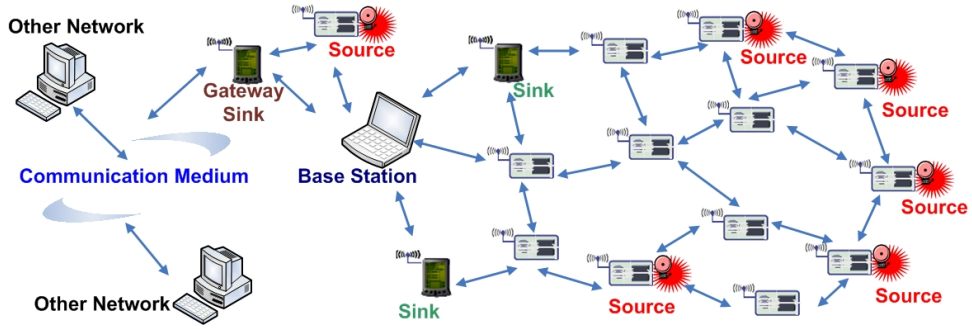


Figure 1.1: Wireless Sensor Network

on the application of the WSN. In some WSNs a sink node acts as *gateway* node. The gateway nodes are connected to external networks (by wireless or wired communication) and the data from the complete WSN is gathered. Figure 1.1 illustrates a typical WSN, showing source, sink, gateway, base station and intermediate nodes. The arrows indicate bidirectional links.

An important consideration in WSN is the *communication medium*. Choices include radio frequency (RF), optical link, infrared, and ultrasound (US). Optical link are appropriate for certain applications like smart dust [1], which contain sand-sized nodes that scavenge energy from their environment. Some networks also use a different frequency optical link called infrared. Both require line of sight between nodes and are influenced by weather conditions. RF on the other hand is widely applicable because it doesn't require line of sight. Finally, both RF and the optical signals fail under water, whilst ultrasound (US) has been successfully used for submarine communications. Communication in all of these media may be adversely effected by the environment in which the WSN operates. As WSNs most often employ RF for communication, the environmental factors like interference (noise) and signal fading can cause substantial message loss.

In a WSN, the nodes need to interact with their environment. Thus WSNs use different *operating systems (OS)* to manage interaction and resources in the nodes. Most notable is the event driven OS, TinyOS [2], which is suitable for minimal hardware resources and concurrency intensive operations. TinyOS is intended to be incorporated into Smart Dust [1] which may become as small in size as dust particles or grains of sand. Some applications are also built on TinyOS e.g. a query processing system for extracting information called TinyDB [3]. OS like PEEROS (Preemptive EYES Real Time Operating System) [4] and Salvo Pumpkin [5] are real

time kernel base operating systems. PEEROS has a preemptive scheduler (relieving the programmer from controlling the hardware), whereas Salvo provides priority-based multitasking, inter-task communication and synchronization. Other operating systems developed for WSN include CMX [6], MagnetOS [7], PalmOS [8], and Mate [9].

1.1.1 Differences Between WSNs and Other Networks

In contrast to wired networks, in WSNs each node may act as a router and WSNs do not require any infrastructure. These properties enable WSNs to monitor data in hostile environments which would be difficult for wired networks. WSNs, if used wisely, can also be deployed in a wilderness for some time without being charged.

Another category of wireless network type is the ad-hoc. These networks may be a set of laptops connected together wirelessly, for a specific purpose. These networks self-configure and they operate without management or infrastructure. A combination of mobiles and PDAs also falls within the description of ad-hoc networks. The WSN [10] is different from traditional ad-hoc networks in several ways e.g. (i) WSN nodes have limited memory, power (nonrenewable), and computational capacities instead of the ad-hoc network's powerful nodes like laptops and PDAs; (ii) energy efficiency is less important for the ad-hoc nodes; (iii) the number of nodes (scalability) in a WSN is much higher; (iv) the nodes are more densely deployed in WSN; (v) WSN always has one or more powerful nodes (base stations) instead of all being the same role (homogeneous) nodes in ad-hoc networks; (vi) nodes in WSNs must cope with frequent topology change due to addition or deletion of the nodes (as a result of node/power failure, intermittent radio interference, and environmental factors); (vii) WSN nodes may fail due to their harsh environment and (viii) the nodes in WSN are usually considered stationary whereas the ad-hoc network nodes may be mobile (MANET).

Although the resources of WSNs have some limitations compared to the ad-hoc network, these limitations do yield certain advantages e.g. (i) dense/large deployment of WSN (a great number of nodes) give a high level of fault tolerance and large area coverage; (ii) the small radio range of WSN means nodes may always be near to their sensed object and thus environmental factors such as interference in the measurement will be small as compared to ad-hoc networks; (iii) high density may enable several nodes to measure the same variable which improves the quality of sensing and may support the authentication of the sensed data. Thus, WSN in most cases provide precision in monitoring as compared to the ad-hoc networks.

1.1.2 Applications of WSN

WSNs have numerous applications [11]. They have been used to measure the temperature, humidity, pressure, noise level, lightening conditions, soil make up, movements (vehicles or living objects) including speed and direction, the presence or absence of certain kinds of objects, and

mechanical stress levels in the structures. The fault tolerance, self organisation and rapid/low cost deployment enables WSN to be used in some military applications. If the nodes are deployed densely and in large numbers, the desired results can still be achieved by WSN, even in the hostile environments and following the destruction of some nodes by the attackers. In traditional networks a minor failure may incapacitate the whole system. WSNs also have been used in environmental, health, commercial and home applications. Some of the potential applications of WSN are categorized below:

- *Military applications* include military command and control; military communication and intelligence; battle damage assessment and surveillance; reconnaissance of opposing forces and terrains; monitoring friendly forces, equipment and ammunition; guidance and navigation systems; and biological and chemical attack detection.

- *Environmental applications* include track the movement of insects, small animals and birds; monitoring environmental conditions affecting crops; irrigation monitoring; earth and environmental monitoring; planet exploration; precision agriculture; forest fire detection; flood detection (rainfall, water level and weather sensors); pollution study and geophysical research.

- *Health applications* include patient monitoring (especially for disabled patients); telemonitoring of human physiological data (help doctors identify predefined symptoms earlier); tracking doctor's location in hospital; monitoring patients (heart rate, blood pressure etc); and drug administration.

- *Commercial/home applications* like environmental control in rooms (air conditioning and heating); remotely controlling commercial/home appliances (VCR, ovens, microwave, vacuum cleaners, refrigerators etc); monitoring product quality in factories; robot control and guidance in automatic manufacturing environments; machine diagnosis (faults/working etc); transportation; preventing car theft; and vehicle tracking on highways and roads.

1.2 Routing and Security in WSN

As WSNs are severely limited with respect to CPU speed, power, memory and bandwidth, innovative techniques are required to overcome these limitations and route data from the source nodes (which sense data from environment) to the sink (or BS). The process of determining the path of data from sources to the destination or sink is called routing. *Routing* techniques are continuously evolving and improving the performance of WSNs. Designing routing protocols is a challenging area [12]. Routing protocols for WSN are quite different from traditional routing protocols. A routing protocol for WSNs must be designed to satisfy properties like connectivity (most nodes must be reachable); coverage (maximum environmental area should be covered);

fault tolerance (WSN should work even following the failure of some nodes); high scalability (the number of nodes may be high); varying density (the number of neighbours of a node may vary widely throughout a network); severe hardware constraints (memory, computation, bandwidth); low cost development; power efficiency; and varying topologies (nodes may be deleted or added at any time). Routing is a central research activity in WSN and many innovative techniques are developing and being improved.

Due to the limitations of nodes, the broadcast nature of transmission, hostile/harsh environment and unattended nature, many attacks are possible on WSNs. An intruder can easily eavesdrop all the broadcast traffic between the nodes. An attacker may modify the messages, inject false messages or later replay the same message. A node can be compromised enabling the dropping of all data packets; the selective forwarding of data messages; spoofed message injection; or presenting multiple identities of the same node. The laptop class attackers can cause unidirectional links or virtual links between long distant nodes. Thus, routing protocols are vulnerable to a number of attacks. Considerable recent research has focussed on making routing more robust against possible interference.

1.3 The Dissertation

This dissertation addresses denial of services (DoS) attacks on WSN routing. A potent technique, called *formal modelling*, is applied to detect the attacks in some published routing protocols. The results achieved from formal evaluation are supported by computer simulation. On the basis of this work a new protocol is developed that is proved to be secure against many DoS attacks using formal modelling. Computer simulation is also undertaken to confirm that the new protocol is scalable.

1.3.1 Motivation and Overview

This research concerns attacks on WSN routing protocols, particularly the attacks which prevent the data from reaching the end points (sinks). These attacks are categorised as denial of services (DoS) attacks [13, 14, 15] and are most challenging to detect and avoid. The term *denial of service* is used in a general sense to mean the adverse effect of any malicious external agent (attacker) on the correct delivery of data from the source nodes to the sink nodes. This research work focuses on the effects of DoS on routing protocols in WSNs. Some examples of DoS attacks are black hole, invisible node attack (INA), hello flood, rushing, sinkhole, spoofing, and wormhole. This research does not address encryption issues; it is assumed that an encryption mechanism is already in place and the attacks specifically related to encryption have been excluded.

Recently, numerous solutions have been presented to detect and avoid the DoS attacks on WSN. Many of these schemes are only designed to address one type of attack while a few schemes

address multiple attacks. Moreover, some of these protocols require specialized hardware such as GPS, directional antennas, etc. Specialized hardware may be available for ad-hoc networks but are not justified for resource constrained WSN nodes deployed in large numbers.

INSENS has been widely regarded [16] as secure against multiple DoS attacks. The earlier version of INSENS [17, 18] however possess some weakness/flaws and an improved version [19] was later presented to solve the hello flood and rushing attacks. The current research however detects some problems in this new version as well and thus proposes a new protocol. An attack 'wormhole' has also long been the center of attention for researchers and solutions presented for the wormhole attack usually require some specialized hardware. A similar kind of attack, INA, is also believed [20] to be possible in all routing protocols. The thesis considers these two attacks and proposes a simplified solution that is possible in WSN nodes not equipped with additional resources.

This research uses an innovative design method of combining a formal modeling and the computer simulation (sometimes hardware implementation as well) to evaluate the routing protocols. Formal models have been used by researchers for quite some time. It has been shown that the formal models can automatically detect the hidden bugs and the worst cases. Researchers have realized that the computer simulation alone is often inadequate for finding errors in the routing protocols. Many bugs in secure schemes were found using formal methods [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. This research uses a formal modelling technique to evaluate different routing protocols and then compare the results with the already published work. Some of the results of the formal modelling were also supported using computer simulation. By use of this formal modelling technique, bugs (vulnerability to attacks) have been found in all the routing protocols considered including INSENS. Therefore, a new protocol that can possibly address those attacks is purposed in this thesis. The new protocol was then rigorously tested by applying the formal modelling techniques. The results were then supported using computer simulation. Computer simulation were also done to check scalability (large node networks) and to quantify the results. For some cases practical hardware implementation was also undertaken to check real world behaviour such as real radio links, fading effects, other practical issues etc.

1.3.2 Problem Statement

The purpose of this research is to explore the following hypothesis: "The application of formal modelling automatically detects the vulnerability of WSN routing protocols to DoS attacks and supports the development of routing protocols resistant to DoS attacks in WSNs"

1.3.3 Main Contributions to Research

The main contributions to research are:

- Formal Specification/definition of recognised WSN attacks. These definitions are more widely applicable to ad-hoc networks and MANETs.

- Development of a Formal Framework to check the vulnerability of different routing protocols. The initial framework was based on the Andel's framework [37]. However, a much improved framework was later developed to accommodate many other routing protocols within the limitation of the state space explosion.

- Formal verification of routing protocols that have not been previously studied, including TinyOS Beaconing, Authentic TinyOS using uTesla, Rumour Routing, LEACH, Direct Diffusion, INSENS, ARRIVE and ARAN.

- An unique and innovative defence against the wormhole attack and INA is presented; it does not require any additional hardware and is suited to WSN nodes.

- The detection of weakness in INSENS, Arrive and ARAN - WSN protocols which are widely regarded as secure.

- Development of a new routing protocol that can work better in the presence of hello flood, rushing, wormhole, black hole, gray hole, sink hole, INA and jamming.

- An innovative design method of using the combination of formal modelling and simulation is presented to evaluate performance of existing and the new protocols against attacks.

1.3.4 Structure of the Dissertation

This dissertation is organised as follows:

Chapter 2 presents a literature review undertaken for this research. It begins by reviewing routing in WSN and then describes recognised attacks/threats to WSN routing and the different secure protocols published for these attacks. The chapter finally addresses some formal modeling techniques applied to WSN, WSN routing and in particular the WSN security against the attacks.

Chapter 3 describes the assumptions adopted for this research. The chapter discusses the method adopted and developed formal framework in detail. Finally, an innovative design method (a combination of formal model-checking and simulation) to address (DoS) attacks on WSN routing is presented which is a contribution to research.

Chapter 4 is dedicated to the formal specifications of WSN and DoS attacks. These specifications are used to define the attacks separately as one generalized attacker model does not have sufficient details and to present specifications of different DoS attackers in a precise and formal manner. This is a contribution to research.

Chapter 5 evaluates some previously published routing protocols, against the WSN attacks, by applying formal modelling. The results were compared with the published work and if

were found to be different, the formal modelling work was augmented by using computer simulation. The protocols considered are TinyOS [15], Authentic TinyOS using μTESLA [38], Rumour Routing [39], LEACH [40], MCF [41], Directed Diffusion [42], LEAP [43], Enhanced INSENS [19], ARRIVE [44] and ARAN [45]. Note that this is a contribution of this research as the formal verification of these routing protocols had not been done before against the attacks like black hole, hello flood, invisible node attack (INA), rushing attack, wormhole etc.

Chapter 6 introduces a new routing protocol, RAEED. The protocol is presented in 3 phases: Key Setup Phase (KSP); Route Setup Phase (RSP) and Data Forwarding Phase (DFP). An innovative design approach of using a combination of formal modelling and simulation has been adopted in this chapter.

Chapter 7 evaluates the new protocol using formal modeling and computer simulation against different attacks such as hello flood attack, rushing attack (KSP); invisible node attack, sinkhole attack, tunnel attack, wormhole (RSP); black hole attack, gray hole attack and jamming attack (DFP). The limitations existing in the new protocol are also discussed in the chapter. The solution presented for wormhole attack and INA is a contribution to research.

Chapter 8 presents some conclusions and proposes future work. The chapter highlights a variety of results obtained in the dissertation from different experiments (formal/simulation), the main achievements obtained, and shortcomings present. Finally, based on the shortcomings identified, the chapter introduces the areas for future investigation and possible research.

Chapter 2

WSN, Security and Formal Modelling

2.1 Introduction

WSNs are composed of small sized, low cost, limited resourced embedded devices (nodes) that communicate wirelessly in multi-hop manner. The communication protocol stack in a WSN is similar to the OSI model (TCP/IP) of the internet. The protocol stack is composed of Physical layer, Link/Medium Access Control (MAC) layer, Network layer, Transport layer and Application layer.

- The *Physical Layer* deals with the hardware issues of nodes such as frequency selection, carrier frequency generation, data encryption and detection and signal modulation etc. The protocols/algorithms of WSNs depend on the requirement of physical layer components [46] e.g. type of micro-controller, type of receivers etc.

- The *Medium Access Layer* (MAC) or *Data Link Layer* is responsible for minimizing the collisions with the neighbor node broadcast; error control; multiplexing of data streams; data frame detection and medium access control (i.e. basic WSNs communication infrastructure; efficient communication resources between the nodes); and message encryption.

- The *Network Layer* is responsible for routing data from the source nodes to the sink nodes. This layer is normally designed by taking into account the resource constraint, scattered/random distribution and power limitations of the nodes.

- The Transport Layer maintains the flow of data when a WSN needs to interact with some external networks or Internet. The communication may be performed by UDP or TCP via internet or satellite.

- The *Application Layer* makes the hardware and software of lower layer transparent to WSN management applications when a WSN interacts with other networks and different

application softwares are used.

As indicated earlier the communication between the nodes is broadcast and in multi-hop manner. An individual node will not be able to transmit the data to the destination directly due to low power used in the transmission. The communication is broadcast thus all nodes near the transmitting node can receive the message. The position of the nodes is unknown before the deployment and the topology of a network might change at any time due to the addition or deletion (power depletion, failure etc) of nodes. The nodes remain unattended once deployed. The total number of nodes including those within the radio range might vary from very few to extremely large in a network. A node may have a number of constraints including memory, computation power, and bandwidth. The biggest issue being that the nodes must consume as less power as possible to give them a long life. These concepts are quite different from traditional networks and thus the process of routing is also different in a WSN. This chapter discusses the issues related to routing (Section 2.2) with the focus given to attacks in routing (Section 2.3) and in particular denial of services attacks (Section 2.4), which is the main aim of this research. Finally the chapter discusses the application of formal modelling on WSN (Section 2.6).

2.2 Routing in WSN

The routing process in WSN involves forwarding the data from the source nodes to the sink nodes. Due to the peculiarity of WSNs, their routing protocols are different from those used in conventional networks. When designing a routing protocol the following features of WSN nodes must be taken into account:

- The *Coverage* which refers to covering all possible area of environment under study with the nodes.
- *Fault Tolerance* which refer to the ability of the WSN to operate reliably without any interruption since some nodes in the network may fail. This failure might be due to physical damage, environmental interferences (affecting the signal propagation), or power depletion. This fault tolerance depends on the environment and varies depending on the type of WSN deployment (home, battlefield, etc)
- *Scalability* means the increase in number of nodes must have no affect on the WSN. The number of nodes may range from few to thousands of nodes. The routing protocol should be designed to accommodate any network size. Since new nodes may be deployed at anytime.
- *Node density* means the number of nodes in a region. The higher the node density, the larger will be the number of neighbors of each node and vice versa. The protocol should

work correctly irrespective of whether the density is high or low.

- *Hardware constraints* of the nodes are also considered when a routing protocol is designed such as small size, extremely low power consumption, low development and production cost, environment adaptability (even to high humidity and temperature), etc.

- The routing protocol must accommodate the *low cost* of WSN nodes. Since a WSN usually consists of a large number of nodes thus the cost of a single node should be very low. The nodes may perform many functions and require built-in ADC, DAC, and other sensors. However, expensive hardware such as GPS should be avoided.

- *Power Consumption* of WSN determines the actual life of a WSN. No matter how many fault tolerance techniques may be present in a WSN, there will always be a point where failure of some nodes will partition the network and thus the WSN as a whole may fail. Since significant power is lost in the message transmission, appropriate effort should be made by a routing protocol to route the data intelligently and consume the least possible power.

- *Connectivity* which ensures the maximum WSN nodes which must be reachable. A WSN topology is mostly undefined and unknown because the nodes may be deployed through various ways. After the node deployment, the topology might change due to possible failure of the deployed nodes or redeployment of the new nodes.

As stated earlier, the routing protocol's design depends mainly on the application at hand. Generally, it is chosen by considering the cost, energy efficiency, latency, distribution density, scalability, quality of service (QoS), or security. Since there exists no protocol which can provide all these features, the main emphasis is often placed on only a few features. This research is based on routing protocols that address the security issues; in particular the attacks on routing that will lead to denial of services (DoS) attacks [13, 14]. Since WSNs are often deployed in hostile environments which requires that the WSN must not only be protected by providing encryption at the link layer, but also some mechanisms at the network layer to avoid DoS attacks. These protections must also consider the limitations of the nodes (Section 1.1.1). Traditional encryption schemes, specially the public key encryptions are not possible in a WSN due to its limitations (mostly modified versions of schemes are used). Moreover, mounting extra hardware or tamper providing resistance mechanisms on the nodes will increase the cost of each node. It is also noteworthy to know that most schemes presented so far address one type of attack and ignore other attacks thus rendering them not feasible in the hostile environments. Although a lot of research work is being done on secure routing protocols but being a novel field plenty of gaps still exists. This research work intends to expose, explore and solve those gaps.

Overall, WSN routing protocols can be classified based on the network structure or the applications it provide. This leads to dividing all WSN routing protocols into one of the following 5 categories namely:

1. *Flat routing* (Flooding [47], Gossiping [48], SPIN [47], TinyOS beaconing protocol [15], Directed Diffusion [42], Rumor routing [39], CADR [49], Energy-aware routing protocol [50], ACQUIRE [51], Gradient-Based Routing [52], SAFE [53], etc)

2. *Hierarchical routing* (LEACH [40], PEGASIS [54], TEEN [55], APTEEN [56], COUGAR [57], etc)

3. *Geo-routing* (GPSR [58], Location Aided Routing [59], DREAM [60], MECN [61], SMECN [62], GAF [63], GEAR [64], GeoGRID [65], GMP [66], etc)

4. *Qos-routing*(MCF [41], SAR [67], SPEED [68], MMSPEED [69], Energy-Aware QoS protocol [70], GRAB [71], Lifetime Routing [72], Upper Bounds Lifetime Protocol [73])

5. *Secure/Robust routing* (Section 2.5).

2.3 Attacks and Security Issues in WSNs

In the recent past, networks relied strongly on physical security e.g. network firewall. The firewall provides access control division between the insecure public network (internet) and secure private internal corporate network. Conversely, the wireless medium is shared and is completely exposed to the outsiders. This makes security infrastructure of the wired network unrealistic to be used because any of the OSI/ISO layers can be attacked by an outsider. The physical layer can be jammed, the link layer coordination packet can be disrupted, and the routing is susceptible to DoS attacks. Also, the transport protocols can suffer targeted attacks (attacks against packets addressed to a specific port) and the applications can be attacked at the application layer. Some of these attacks at different layers are explained in [74]. In WSNs, a message usually takes multiple hops before arriving at the destination. The broadcast communication, unattended deployment and hostile environment make WSN nodes vulnerable to many attacks. This research work is aimed at addressing attacks at Network layer or routing.

All attacks on WSN routing can be categorized as 'passive' or 'active'(Figure 2.1). In a **passive attack**, the attacker passively observes (eavesdrops) ongoing communication and does not send any message. Examples are eavesdropping attack and traffic analysis attack. In *eavesdropping attack*, an attacker observes the sensed data or application specific content of messages. The *traffic analysis attack* involves measurement of the angle of arrival of a packet and the signal strength to locate and destroy the important nodes such as source and BS. All protocols that provide some kind of cryptography (Section 2.5.6) can address eavesdropping

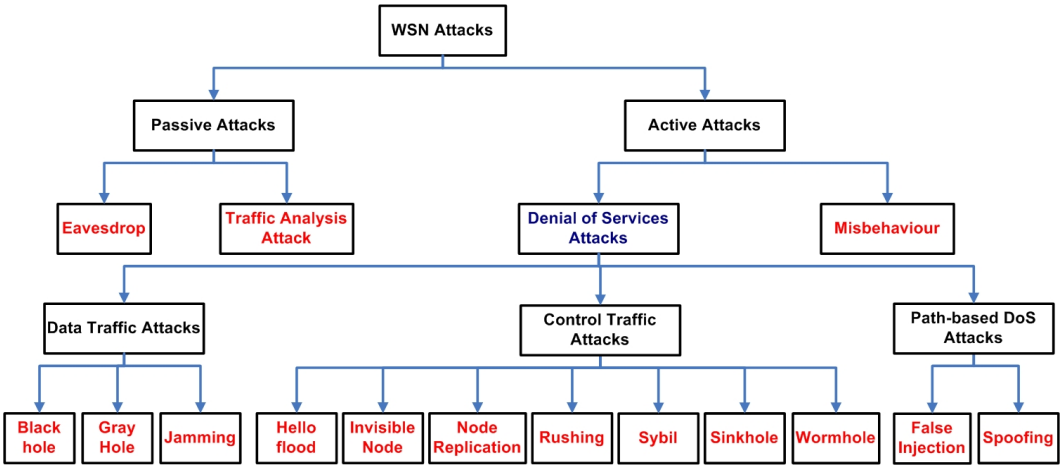


Figure 2.1: Attacks in WSN

attacks. Some solutions for traffic analysis attack are described in [75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85].

In an **active attack** an attacker observes the communication and involves in communication by injecting the packets in the network. The active attack is further classified into 'Misbehavior' and 'Denial of Service (DoS)'. *Misbehaviour* is an attack in which an attacker behaves selfishly by deviating from the legitimate protocol operations. Some of the solutions presented to address misbehaviour attack are described in [86, 87, 88, 89, 90]. In *Denial of Service* attacks [13, 13], the attacker does not directly manipulate the protocol parameters and instead aims at indirect benefits by unconditionally disrupting the network operation.

2.4 Denial of Service Attacks

The aim of DoS attacks is to disrupt the routing and prevent data generated from the source nodes to reach the sink or destination nodes. DoS attacks can further be classified into 'Control traffic attacks', 'Data traffic attacks' and 'Path based DoS (PDoS) attacks'. The *Control traffic attacks* are targeted at monitoring the liveness of the nodes, topology discovery, and the disrupting routes. For example hello flood, wormhole attack, invisible node attack, sybil attack, sinkhole attack and rushing flood attack come under this category. In *Data traffic attacks*, an attacker drops or attempts to drop data packets e.g. black hole attack, selective forwarding and jamming. The Control attacks are more potent than Data attacks because they are used to disrupt the functionality of the routing protocol and create opportunities to launch Data

traffic attacks. Moreover, due to limited resources, the Control attacks are more challenging in WSNs than in ad hoc wireless and wired networks. Finally in the *PDoS attack*, an attacker overwhelms distant nodes by flooding a multihop end-to-end communication path using replayed or false messages [91]. Spoofing attack and false injection attack come under this category. A strong encryption scheme can prevent PDoS attacks. The research is aimed towards proposing solutions to DoS attacks that prevail even in the presence of strong encryption schemes. A brief introduction to all these DoS attacks and some of the notable solutions are explained below:

2.4.1 Spoofing Attack

Spoofing is based on altering or replaying the routing information (data, beacon or acknowledgements) exchanged between the nodes. Spoofing leads to many routing problems e.g. routing loops are created, routes can be extended or shortened, the network traffic is attracted or repelled, the generation of false/error messages which might lead to network partition (forged routing packets to prevent one set of nodes from reaching another) and an increase of an end-to-end latency by adding the virtual nodes to the route. Spoofing attacks can only be avoided by using some encryption techniques. Therefore, all protocols that provide some cryptographic mechanisms (Section 2.5.6) are potential solutions to spoofing attack. Some other DoS attacks like framing and stealthy attack also come under spoofing.

'Framing' is a form of spoofing in which an attacker maligns another node without provocation or a justified cause. The malicious nodes thus frame the legitimate nodes. This attack is common in protocols where the nodes cast votes to incriminate suspect nodes [92].

Stealthy attack is another extension of spoofing in which an attacker's goal is to allow acceptance of false aggregation results, which are significantly different from the true results determined by the measured values, as well as not being detected by the user. Secure Information Aggregation (SIA) [93] presents the aggregate-commit-prove framework for designing secure data aggregation protocols.

2.4.2 False Injection Attack

False Injection attack is similar to spoofing attack. The difference is that the attacker injects extra messages (data or control) into the network instead of modifying the received messages before the forwarding process. The aim is to extract the maximum resources from the legitimate nodes i.e. exhausting the limited energy, memory, and CPU of resource-limited sensor nodes, to consume the communication bandwidth and sometimes to cause routing loops. By flooding the replayed or spurious messages, an attacker can even overwhelm the nodes a long distance away from it. To defend against this attack, a node must detect spurious and relayed messages and reject them. Possible solutions to this attack are separate shared key schemes (Enhanced INSENS [19], LEAP [43]); Protocols using message authentication code (SEF [94], Interleaved key

scheme [95]) and schemes using one-way hash chains [96] (Enhanced INSENS [19], Ariadne [97], PDoS defense scheme [91]).

2.4.3 Hello Flood Attack

Some routing protocols require the nodes to announce themselves (to neighbours) using a 'Hello' message. This message enables the receiver nodes to assume that the sender is within their radio range. However, this leads to hello flood attacks [15], in which a laptop-class attacker broadcasts the routing/other information with large enough transmission power to convince many or all the nodes in the network that the adversary is a genuine neighbour. The attack thus causes unidirectional links between the attacker and the legitimate nodes. Some notable solutions for this attack are Hello Flood Attack and Defense scheme [98], LEAP [43] and Enhanced INSENS [19].

2.4.4 Wormhole Attack and Invisible Node Attack (INA)

In a wormhole attack [99], an attacker or malicious node records the packets (or bits) at one location in the network and tunnels them to another malicious node at a distant location, which then replays (retransmits) them locally. The tunnel can be established in many different ways, such as through an out-of-band hidden channel (e.g., a wired link or using a different radio frequency), a packet encapsulation, or high powered transmission. As the aim of the tunnel is to enable the tunneled message to arrive either sooner or with fewer hops compared to the packets transmitted over legitimate multihop nodes, so the hidden channel attracts legitimate traffic. Packet encapsulation is less effective because the same channel must be used and a few nodes have to be compromised for its success. Some researchers also call it as *Tunnel attack* and encryption can solve some of these problems. This research's wormhole definition states that a wormhole attack is possible only using hidden channels and invisible nodes, which is much more potent than the tunneling attack.

Invisible Node Attack (INA) [20] can be considered as a simple form of a wormhole attack in which a single malicious node simply retransmits whatever messages it receives during the route discovery process, without adding itself to the routing path. The node remains invisible to other nodes and creates virtual links between 2-hop unconnected nodes. This attack is less potent than a wormhole attack as the attacker uses the same channel. The sender nodes can detect INA as they will observe receiving the same packet which they transmitted earlier which is not possible in the wormhole. As INA is a variation of a wormhole attack, some of the solutions for the wormhole can also be used for INA.

The nodes within the radio range of both wormhole tunnel nodes (or a single invisible node) believe that they are all neighbours. In practice there is only a virtual connection between them. A wormhole tunnel may be very useful if used for forwarding all the packets.

However, the attackers may drop the messages at any time especially the data packets. This leads to black hole (drop all data packets) or gray hole (drop some data packets) attacks. In wormhole, the attacker does not need to know the encryption mechanism or any keys and the attack is possible even if the attacker has not compromised any nodes or even if the encryption scheme provides authenticity and confidentiality in all communication. Hence wormhole is rather difficult to detect. Finally, the attacker does not need to allocate any computational resources to compromise the communication, thus rendering the wormhole attack to be launched very easily. Many solutions have been presented to solve the wormhole attack. However some solutions like the RF fingerprinting [100] and Directional Antennas approach [101] are not feasible in resource constrained WSN node. Some of the other wormhole solutions are described bellow:

- Some schemes use a *specialized hardware* to solve the wormhole attack. *SECTOR* [102] uses a special hardware transceiver module and an authenticated distance-bounding protocol to estimate the distance between two neighbours. There are some disadvantages though. Apart from requiring the nodes to measure the local timings with nanosecond precision (very difficult in WSNs), the random delays introduced by MAC layer jeopardize this method of protection.

- Some techniques are centralized and use *statistical methods* to detect wormhole after gathering information about the node neighbourhood and sending this data to BS. The BS then checks the probability of the wormhole attack using the data received and making a statistical graph of the neighbours of all the nodes. Notable examples are *Statistical Graph Test* [103], *Acoustic signals under water scheme* [104] and *Connectivity Information Knowledge scheme* [105]. Shortcomings in these schemes are that (i) they can only detect the presence not the location of wormhole, (ii) are successful only when the radius of the wormhole is comparable to the radio range of the sensors, and (iii) waste bandwidth (overhead of sending neighbor information to BS). Moreover the acoustic signals scheme [104] can have an error of up to 40% of the radio range.

- Some protocols use *local monitoring* to detect, isolate, remove or at least mitigate the wormhole attack. Notable examples are *LITEWORP* [106, 107] and *ODSBR* [108]. Some shortcomings in these schemes are that (i) a number of data messages are lost before detection, (ii) large number of shortest paths will be through the wormhole so causing more data loss and thus increasing time taken to detect wormhole and (iii) a lot of overhead as every data packet needs to be acknowledged by the destination.

- Some schemes use the *geographical position* of the nodes to detect wormhole attack. In these cases either the nodes are equipped with GPS or their positions are known through some secure way. Two important schemes are *TIK* [109] (Geographical Packet Leashes

and Temporal Packet Leashes) and *COTA* [110]. The disadvantages of both TIK approaches are that they require either location information of each node (GPS) or tight clock synchronization between the nodes (nanosecond precision clock is required which is very hard in WSN). On the other hand, apart from requiring the location information, COTA keeps a constant space for every node on the path and the computation overhead increases linearly to the number of detection packets.

- Some schemes use *'Time of Flight'* to detect a wormhole attack. Examples are RF based solutions [111, 103, 112], *TrueLink* [113], Ultrasound (US) based scheme [114] and Ranging protocol using US [115, 116]. Problems with these schemes are that (i) they require tight clock synchronization (nanosecond precision clocks) (ii) and light speed propagation of RF waves. Additional disadvantages in US-based solution are that (i) the distance measurement is not secure and (ii) an attacker can decrease the distance by relaying a slow US signal over a fast relay link.

- Some schemes use a few special nodes (Guards) to monitor other nodes e.g. *Trusted Specialized Guards* [99]. The guards have a larger radio range than the other nodes and are the only nodes equipped with GPS thus helping other nodes to get their location information. The disadvantages in these schemes are that (i) the guards are distinguished nodes (high resource) in the network that differ from the regular nodes, (ii) requires extremely tight time synchronization (not easily feasible in WSN), (iii) placement of guards is important (very hard in random deployment) and (iv) and the existing protocols cannot be easily modified using that technique.

2.4.5 Sinkhole Attack

In a sinkhole attack, the intruder attracts all the data traffic towards itself and does not forward it further. A sinkhole can be (i) a laptop with high range that can provide less hop paths to other nodes, (ii) a spoofed message indicating a node is near to BS or a BS, (iii) a wormhole providing low latency and less hop distance links or (iv) a hello flood attack providing unidirectional links to unreachable nodes. A few sinkhole solutions are described in [117, 118].

2.4.6 Sybil Attack

In a sybil attack [119, 120] a single attacker node presents multiple identities to other nodes in the network. A sybil attack will thus make a node to be present in more than one place at a time. It is stated in [15] that a sybil attack can disturb a multi-path routing where seemingly disjoint paths could in fact go through a single malicious node presenting several sybil identities. Geographic routing is also vulnerable to sybil attack since, instead of having one set of coordinates, a sybil node could claim having multiple coordinates and thus appear at more than one place simultaneously in other nodes. Typical sybil attack solutions are *Enhanced*

INSENS [19] (develops a pair-wise key between nodes in KSP and marks secure neighbours using bidirectional verifications), *Radio resource testing scheme* [120] (multiple channel broadcast for individual neighbour) and *Sybil-free pseudonyms* [121] (self-certified cryptographic IDs).

A similar attack is *Clone* or *Node Replication* attack, in which the attacker present same ID for multiple physical nodes. Some of the solutions for clone attack are [122, 123].

2.4.7 Rushing Flood Attack

Most routing protocols require flooding of some route information at some stage. In order to flood the duplicate information, the nodes suppress any duplicate message. This duplicate suppression property leads to rushing attack [124], in which the attacker disseminates the route information quickly (via hello flood/wormhole) throughout the network. This leads to nodes receiving incorrect information, and dropping the legitimate route information received later as a result of the duplicate suppression. The rushing attack can be made more effective when the attackers near to source of legitimate information jams the surrounding nodes after receiving the information. This further prevents the legitimate information to reach the nodes timely. On the other hand a fast link between the attackers will rush data enabling the information to reach farther end nodes. This results in increasing the chances of rejecting legitimate messages due to the duplicate suppression. Some of the rushing attack solutions are *Enhanced INSENS* [19], *Rushing Attack Prevention* (RAP) [124], *Secure Implicit Sampling scheme* (SIS [125] and *ODSBR protocol* [126, 108].

2.4.8 Jamming Attack

A possible attack on WSNs relates to create a noise or collisions in particular areas where all the nodes are jammed. Jamming is usually aimed at the physical layer by broadcasting high transmission power signal to corrupt a communication link or an area. Jamming is also an attack to the routing as it makes all traffic in that area useless and prevents data from reaching the destination. This type of attack is very effective because no special hardware or extra cost is needed by the attacker and it can be launched by listening and broadcasting the same frequency the network is using. Spread spectrum modulation [127], or directional antennas have been extensively studied to improve the resistance to physical jamming. But it can be too much energy-consuming and is not suitable for WSNs. The works carried out in [14, 128, 129, 130, 131, 132, 133, 134] involve different research works on jamming involving network and MAC layer. Some of the jamming solutions in routing protocols are *Enhanced INSENS* [19], *JAM* [135], *Wormhole base Anti-Jamming Solution* [136], *Channel hopping schemes* [137, 138], *Switch frequency scheme* [139], *MMSN* [140], *DEEJAM* [141], *Jamming monitor scheme* [142], *Denial of message attacks scheme* [143], and *Link-layer denial of service scheme* [144].

2.4.9 Black hole Attack and Selective Forwarding Attack

In a black hole attack a malicious node discards any data message it receives after joining the network. The malicious node might join the network via a virtual (INA or wormhole) or real (node capture or encryption failure) connection. If a node discards all the data messages, the neighbours might realize that the node is dead and thus find an alternative path. A better attack would be to forward a small percentage of the messages and drop the remaining. This attack is called *selective forwarding* or *gray hole*.

Both these attacks might be launched in many ways. A malicious node on the data route path intentionally drops, delays or alters the data traffic passing through it. Moreover, the malicious node can advertise itself (impersonate) as having the shortest path and cause data traffic to route in a wrong way (sinkhole attack). The solution to node *dropping data* is to deploy *neighbourhood watch* i.e. observe that the next node on the path forwards the data as expected. However, a more potent form of attack called 'colluding black hole' can overcome this mechanism. In a colluding attack multiple black hole attackers work together and thus might route data between themselves instead of forwarding to the required route. Solution to the *problem of attractive advertising* is to solve wormhole/INA and making route decisions locally. A version of black hole on aggregation protocols is called as '*denial of message*' attack by [125]. In this attack, an intruder drops other nodes readings, or deprives other nodes from receiving broadcast messages of the BS (by dropping them). Some of the notable black hole/gray hole solutions are explained next:

- Some schemes use neighbour transmissions and node's neighbourhood link estimates to avoid black hole attack. Examples are *REWARD* [145], *Mint* protocol [146], *Watchdog and Pathrater scheme* [147], *Passive Trust building scheme*[148], *DICAS* [149], *Parent Monitor scheme* [150], etc. Some of the disadvantages are that (i) they are vulnerable to blacklisting (framing attack), (ii) do not work correctly in the presence of colluding black holes, (iii) can have a high error rate due to RF noise/collisions in the wireless channel, (iv) overhearing does not always work (collisions, weak signals etc), (v) in some schemes (Parent Monitor scheme) lot of energy is consumed in probing, (vi) sometimes (Mint protocol) nodes need to periodically broadcast its routing information (communication overhead) and (vii) each node needs to maintain a table which contains its neighbour nodes routing information (storage overhead).

- Some schemes require end-to-end acknowledgments (ACK) from the destination/sink nodes to detect the presence of any black hole attack. Examples are *ODSBR* [126, 108], Symmetric key based solution [151], *Secure Message Transmission (SMT)* protocol [152], *Secure Implicit Sampling (SIS)* [125] etc. Some of the shortcomings in these schemes are that (i) they come at the expense of extra bandwidth usage and node energy because of ACK messages, (ii) sometimes they require more time to detect black holes because one

path can only be checked at a time, (iii) the source nodes must receive unmodified ACK from the destinations that cannot be guaranteed in case any node has been compromised in between, (iv) they can only detect presence of black hole in a path and not the exact attacker and (v) some schemes (SRP) are vulnerable to rushing attack.

- Some schemes choose a forwarding node for data with some probability and forward single/multiple copies of the same data in multiple-paths e.g. *ARRIVE* [44]. The disadvantages of such schemes are (i) unidirectional links and hidden terminal problems may cause the passively participating nodes to act erroneously, (ii) passive forwarding adds extra communication overheads and (iii) probability can never guaranty data is regenerated every time it is lost.

- Some schemes use statistical methods or the nodes convey the topology of network to the BS to avoid black hole attack. Examples are *Monitor Topology by base station* [153], *LITEWORP* [106, 107]. Disadvantages in these schemes are (i) most of times they can only detect the presence not the location of attacker and (ii) waste bandwidth (overhead of sending neighbour information to BS).

- Some schemes use a detection mechanism to avoid black hole attack. Sometimes, a node is elected periodically as an agent (cluster head) to detect intrusions for all nodes in the cluster e.g. *Intrusion Detection System (IDS)* [154]. Some of the shortcomings of these schemes are that (i) the cluster head can be compromised and (ii) attacks on cluster heads cannot be detected. A similar technique used by some schemes e.g. *SODESN* [155] is to use a learning method to train the nodes for fault detection and misbehaving nodes (malicious and non-malicious) during local communication between nodes. Some disadvantages in these schemes are that (i) lot of time and messages are wasted before the attack is detected and (ii) colluding black hole attack is possible.

- The schemes used to discover the dead or faulty nodes in the network can also be used as a solution for black hole attack. Examples are SPINS (Security Protocols for Sensor Networks) [38], centralized schemes that use BS [156, 157]; neighborhood listening schemes [92, 147], neighbours collaboration schemes [158, 159, 160, 161] and schemes indicating energy depletion like residual energy scan (eScan) [162]. Finally the protocols designed for misbehaving nodes can also be used as a solution for black hole.

2.5 Classification of Secure/Robust Routing Protocols

Although a number of schemes were proposed for different WSN attacks, this thesis categorizes the solutions into 6 different classes or mechanisms namely protocols using: multiple-paths, probabilistic path selection, overhear neighbour communication, specialized hardware, topology

mapping and cryptographic techniques. Researchers in some cases have not focussed on one of the mechanism. In many cases they used a combination of one or more mechanisms to provide a solution to WSN attacks. Usually the mechanism relies on one of these 6 categories to provide security. A brief description of these techniques is given below:

2.5.1 Protocols using Multiple-paths

Some protocols employ path diversity techniques to increase the route robustness, reliable data delivery and to avoid attacks. In this case multipath routes are first discovered and data is then routed to these independent paths along with the primary path to provide data transmission redundancy between the sources and sink. There are many ways in which data can be sent in multiple paths. Some schemes send multiple copies of the same data along the different paths, while others fragment data into packets and send these fragments in multiple paths (load balancing). The load balancing is achieved in such a way that even if some of the sub-packets are lost the original data message can be reconstructed.

Multiple-path routings can either be disjoint or braided multi-paths. In *disjoint multi-path routing*, all nodes in separate routing paths are different. Advantages are that a failure of a primary path will not effect the overall transmission but at the expense of longer latency and more energy consumption. Disjoint path are more problematic in lower density networks (longer and high cost paths) as compared to high density networks, where many routes are available. In *braided multi-path routing*, the nodes in the path need not be disjoint from the primary path and thus some nodes share multiple paths. Advantages are low latency and better energy consumption than disjoint multi-path counterpart. However a failure of any common nodes due to attack/power failure will fail more than one path.

The main and obvious drawback of multipath routing is the overall increase of the traffic which is proportional to the number of multiple paths adopted. Moreover some attacks such as sybil attack can lower the effect of multiple paths by presenting multiple identities. The most effective attack against multipath routing is hello flood, wormhole and INA that create virtual links between the nodes enabling false routes.

Some notable examples of disjoint multiple path schemes are basic INSENS [17, 18], enhanced INSENS [19], Directed Diffusion [42], Split Multipath Routing (SMR) [163], ARRIVE [44], Temporally Ordered Routing Algorithm (TORA) [164] etc. Examples of braided multipath schemes are Braided multi-path routing [165], Meshed Multi-Path Routing (MPR) [166]. Some famous ad-hoc network multipath schemes are Dynamic Source Routing (DSR) [167] and its extensions e.g. [168], Multipath On-Demand Algorithm (MDR) [168]. An example of scheme using load balancing technique is [169].

2.5.2 Probabilistic Path Selection Protocols

Some schemes select the routing path probabilistically. A sender selects one of its neighbouring node with some probability. Since, there exists no fixed path to forward the data, an attacker cannot block all routes to BS. This results in the routes taking longer time and data loss is still possible with some probability. Some examples are ARRIVE [44] and Rumor routing [39].

2.5.3 Protocols that Overhear Neighbor Communication

Some protocols utilize the broadcast nature of a WSN communication to overhear neighboring communication to detect misbehaving nodes. This mechanism is simple to implement. Extra storage space might be required if the nodes are required to save a log book of their neighbour's performance. Moreover the scheme fails if some links are unidirectional. Some important schemes using this mechanism are Watchdog [147], Passive Trust building [148], DICAS [149], Parent Monitor scheme [150], SODESN [155]; ad-hoc network solutions like CONFIDATNT protocol [87], Passive Trust building scheme [148]; and MANET solutions like Intrusion Detection System (IDS) [154], Lightweight Robust Routing [88] and Cooperation protocols [89, 90].

These schemes have some limitations. For example, they are vulnerable to blacklisting or framing attack (Section 2.4.1), in which a malicious node claims legitimate nodes to be the attacker. In addition most of the schemes fail in the presence of colluding black holes, where multiple black hole neighbouring nodes work in collaboration. Also, the schemes can have high error rate due to collisions in the wireless channel and overhearing does not always work because of collisions and weak signals.

2.5.4 Protocols using Specialized Hardware

This category consists of protocols that use either tight time synchronization, location awareness through GPS based hardware or directional antennas. These schemes are mostly used to detect and solve wormhole attack. Schemes using tamper resistance hardware also come under this category. As such these techniques are very costly they are not feasible in cost effective resource constrained WSNs.

Some examples using special hardware are the RF-fingerprinting scheme [100] , directional antennas approach [101] and SECTOR [102]. Examples of temper resistance schemes are Security-aware Ad-hoc Routing (SAR) [170] etc. Notable schemes using GPS are Geographical packet leashes scheme in TIK [109] and Trusted Specialized Guards [99]. Important examples using tight clock synchronization are Temporal packet leashes scheme in TIK [109], TrueLink [113],and [103]. Some schemes use a mix of GPS, tight timers or extra hardware e.g. Secure Location US based scheme [114] (GPS and US generator), Ranging Protocol [115, 116] (Tight clock and US generator) etc.

2.5.5 Topology Mapping Protocols

In topology mapping schemes the whole topology of the network is detected usually at the BS. This scheme uses a message passing between different nodes and conveying the information to the BS. Thus, the raw picture of a network can be developed and some attacks such as wormhole, black hole etc can be addressed. Potential drawbacks of these schemes are that they are not scalable and require high traffic overheads. Examples are Basic INSENS [17, 18], Statistical Graph Test [103], MDS-VOW [104] (under water scheme using delays of acoustic signals), Connectivity Information Knowledge [105] and Monitor Topology by base station) [153].

2.5.6 Protocols using Cryptographic Techniques

Cryptographic techniques have been used as building blocks for the security of the protocols. By using simple keys the data can be encrypted at the source and later decrypted at the destination thus preventing malicious nodes from reading the data. Using cryptographic techniques in WSNs are quite different from traditional wired and ad-hoc networks. The cryptographic schemes in WSN can be further divided into 3 major categories:

- Symmetric Key cryptography (SKC) solutions like TinySec [171], ZigBee [172], SPINS [38], MiniSec [173], SEF [94], LEAP [43], LKHW [174], LIGER [175], Hetrogenous SKC solutions [176, 177] etc

- One-way hash chains solutions like Sluice [178], SIS [125] , INSENS [17, 18] , Enhanced INSENS [19] , Path base DoS solution [91], SLIMCAST [179], STAPLE [180], Hop-by-Hop Authentication scheme [95], SDAP [181], SEF [94] etc

- Public Key Cryptography (PKC) solutions like RSA based solutions [182, 183, 184]), Elliptic Curve Cryptosystem (ECC) schemes [185, 186, 187, 188, 189, 190], TinyPBC [191], DoS Resistance PKC [192], Innovative Signature Algorithms [193], Energy Efficient Security [194], One-Time Signature scheme [195] etc.

In WSNs, the symmetric key is preferred over the public-key cryptography (PKC) because nodes have insufficient resources for PKC. The symmetric-key ciphers and cryptographic hash functions are cheaper and faster (orders of magnitude) and data packets in sensor networks are generally small.

2.6 Formal Modelling and WSN

Formal models translate the state diagrams of a program to formal mathematical models. It then asserts to negate the formulas describing the desired properties. In case an assertion is violated it generates a trace to show as to why the assertion is negated. Computer simulation can only execute one of all the possible program traces. Formal modelling on the other hand

can check all the possible executions. The computer simulations thus may or may not check the desired properties of a program. General purpose model checking tools e.g. Spin [47, 196], Uppaal [197], PRISM [198, 199, 200] allow to verify not only the functional properties of a system (e.g. Spin), but also the performance of real-time system (e.g. Uppaal). For example PRISM is a probabilistic model checker, used for formal modelling and analysis of systems which exhibit random or probabilistic behaviour. The fundamental principal underlying a model-checking approach is that of exhaustive search, implying that a "model checked system" is guaranteed to satisfy the specified properties. Some tools such as Casper/FDR2 toolbox and AVISPA are used to verify the authentication issues in protocols. Of all these modelling techniques, Uppaal is extensively used for formal modelling of Real-time Control Network [201, 202].

2.6.1 Formal Modelling in WSNs

The use of formal modelling in WSN community is becoming common now. Fehnker at. el [203] use a graphical specification style to enable the study of the effect of topologies in the performance analysis. GLONEMO [204] is an approach presented for the formal modelling and analysis of ad-hoc sensor networks, at various levels of abstraction, e.g. the hardware that implements a single node; the protocol layers; the application code; and an abstract model of the physical environment. Frederiksen et. al [205] used the Uppaal framework to analyze the energy consumption in WSNs. A state-oriented model of ad-hoc network nodes is presented in [206]. Kwiatkowska et al [207] used PRISM to analyze the randomized back-off procedure in the 802.11 wireless protocols. The work in [208] verifies the correctness of the WSN application focusing on concurrency using Spin. The work in [209] presents a theory of calculus for WSN that combines a network layer of nodes with a local object model to describe the nodes. A clock synchronization algorithm for link layer (TDMA) protocol of WSN 'gMAC protocol' is modelled using Uppaal in [210, 211].

2.6.2 Formal Modelling of Routing Protocols

It has been realized by researchers that computer simulation alone is inadequate in checking a routing protocol and the development of formal models to check different aspects of routing protocols is now becoming common. Many wireless routing protocols (WSN and ad-hoc networks) have been verified using formal modelling. Camara et. al [212] applied formal methods (Spin) to verify the design bugs in protocols for MANET networks such as LAR, DREAM and OLSR. Fehnker et. al [213] used Uppaal to model and verify the link layer protocol LMAC. Henderson et. al [214] used Uppaal to identify flaws in the MCF protocol. The work in [215, 216, 217] evaluate different aspects of flooding in WSN using PRISM. The work in [215] also considers aspects of gossiping protocols. The work in [218, 219] focuses on improving the effectiveness of data dissemination and gathering by optimizing tree-based routing topologies. Krishnamachari

et al [219] studied the performance of different data aggregation algorithms for different network topologies. Nair et al [220] extends the formal model in [215] to include the S-MAC broadcast and unicast. They later use TLA [221, 222] to write specifications of the diffusion family of data dissemination and gathering protocols.

2.6.3 Formal Modelling and Security in WSN

Some important work in WSN security schemes using formal modelling are:

- TinySec [171]and LEAP [43] were modelled in [21] using high level formal language HLPSL. The authors found two attacks (i) a man-in-the-middle attack and (ii) a type flaw attack which shows that confidentiality is compromised. They later analyzed SNEP [22] which is the base component of the security protocol SPINS [38]. By using a formal analysis it disclosed an attack 'false request message from an intruder'.

- The strand spaces model [23] was developed for the formal verification of key exchange protocols. Routing security is defined in terms of a safety (discovered routes do not contain adversarial nodes) and a liveness (it is possible to discover Routes) property. A similar work was presented later in [24] which proposed formal model for ad hoc routing protocols and insider attacks.

- The work in [25], uses a formal method CPAL-ES, to analyze an ad-hoc protocol SRP [223]. It discovers attacks such as wormhole etc.

- The work in [224] articulates a formal security framework which is used for the analysis of protocols for constrained systems applications involving MANET and RFID systems.

- Slede [26, 27, 28] is a framework developed using Spin for automatic verification of WSN security protocol implementations. It extracts models from the protocol implementations and verifies them against generated intruder models. In the case of a property violation, Slede translates the counterexample back to a domain language (nesC). Using this technique they verified μTESLA [38] and LEAP [43].

- A flexible and mathematically rigorous modeling framework (ABV model) was presented by Gergely Acs et. al [29] for analyzing the security of ad-hoc routing protocols in [30, 31, 32] and WSN routing protocols in [33, 34] respectively. They found attacks in both versions of Ariadne ([97, 225]), SAODV [226] and ARAN [45]). They updated their attacker model in each subsequent work. They also developed an extension of Ariadne protocol called endairA and verified it using the mathematical model. However, the model has some weaknesses. Liqiang et al [35] claim that ABV model has incorrectly merged the adjacent adversarial nodes, has improperly defined the system state of the model, and has flaws in the proof for ARAN protocol. Moreover, Burmester et al [36] claim that

the security proof for endairA is inadequate and the model described in [31] has flaws. Finally, Andel using Spin model checker proposed in [37] confirmed that author's claim in [31] that false paths should not be successfully returned to the initiator with the correct appended signatures is incorrect. This puts the credibility of security model in doubt.

- T. Andell et. al [37] used Spin to analyze the security properties in the route discovery phase for on-demand source routing protocols that were the extensions of ad hoc routing protocols for DSR (Ardiadne and endairA). By using their automated security evaluation process, they were able to produce and analyze all topologies for a network size of up to 5 nodes. They were able to detect some flaws in both protocols using Spin tools.

2.7 Chapter Summary

This chapter has discussed the reasons why routing protocols are different in WSNs. The chapter later classified the possible attacks in WSN into two groups passive and active. Eavesdropping and traffic analysis are categorized as passive attacks. Active attacks are further classified as misbehaviour and DoS attacks. The DoS attacks, their existing solutions and shortcomings present in these solutions were briefly discussed. The chapter then classified the secure protocols into different groups namely multiple path, probabilistic path, overhear neighbour, specialized hardware, topology mapping and cryptographic based solutions. Finally the formal modelling was discussed in the chapter and some related work in WSN applications, WSN routing and WSN security protocols was described briefly.

Chapter 3

Proposed Formal Specifications of WSNs and Attacks

3.1 Introduction and Motivation

This chapter discusses formal definitions of WSNs in general and different attackers used in the research. The chapter is organized in two main parts. Section 3.3 presents formal building blocks necessary to define DoS attacks. This includes the formal specifications of basic components such as nodes, messages etc as well as basic operations in WSN such as receive, transmit, links, cryptography etc. Section 3.4 formally defines the attacker models considered in the research. The specifications of each attack is written in Z formal mathematical notation. This formal specification is a contribution to research as this concept is novel.

These definitions of attacks are used later to describe attacker models within the formal framework.

3.2 Motivation

This research is based on applying exhaustive formal modelling techniques on different routing protocols to expose their susceptibility to DoS attacks. Before applying formal modelling it was observed that the definitions of attacks in the literature are somewhat vague and ambiguous. For example, it was found that different authors give their own informal definitions of attacks; these are not always consistent. It was realized that there is a strong need of defining the attacks in a concise manner. The specifications were first written in TLA [221, 222] since some WSN specifications have been so far written in this formalisation [220]. However, it was found that TLA has some shortcomings such as type checking etc. So the specifications were later rewritten in Z.

Note that this work is different from exhaustive model checking which is addressed in Chapter 4. The aim here is mainly to define an attacker's specifications in a concise and formal way

rather than to perform exhaustive checking. These definitions of attacks are used later to describe attacker models within the formal framework. These precise definitions of attacks are then modelled during analysis/evaluation of the published protocols such as TinyOS Beaconing, MCF, LEACH, Rumour Routing, Directed Diffusion, Authentic TinyOS with μTESLA, ARAN, Arrive, INSENS in Chapter 4 and the newly developed protocol RAEED (Chapter 6). The attacker models in all these protocols are developed by employing the definitions of attacks presented in Section 3.4. Note that the formal framework presented later in Section 4.3 requires semi formal or formal definitions to implement all sub models. This is the main reason for employing semi formal notations of different protocols to model the nodes and the BS; while the formal specifications presented in this chapter are employed for the attacker models. These DoS attacks, to the best of knowledge, have not been defined before, thus making it a contribution to WSN research. This can be useful to researchers in the future because they can consult these definitions, rather than the vague informal definitions, to describe solutions for DoS attacks.

3.3 Proposed Formal Specifications for WSN

3.3.1 Basic Definitions

Some basic parameters need to be defined before defining DoS attacks. We define `KEY` as a special variable of alphanumeric type and `FLAG` of boolean type which can only have two values `TRUE` or `FALSE` and used throughout the specifications to set or clear a `FLAG`. We define some constants: the maximum number of nodes in the network and maximum data of 1000 and 2048 (11 bits), respectively. These constants are defined to limit the maximum value of the node ID and the data size in a message.

Since the definitions are made at the abstract level, a message needs to be defined first. It is composed of Message Sender, Message Destination and Message Data. The sender and destination are integers with values ranging from 0 to `MAXNODE`, indicating node IDs. The message data must have a value up to `MAXDATASIZE`. In the case of broadcast message the `MsgDestination` value is ignored.

$$Message \mathrel{\widehat=} [MsgData : 0..MAXDATASIZE;\ MsgSender, MsgDestination : 0..MAXNODE]$$

We then define a schema, `InitMessage`, which initializes the message by assigning to it the initial value before writing a theorem to state that there exist a message satisfying the initial conditions of a message. A `Node` is then defined as follows:

$$Node \mathrel{\widehat=} [NodeIsSink : BOOL;\ NodeID : 0..MAXNODE]$$

Note that it contains a `FLAG` indicating that it is a sink (base station) or a normal node; and a node ID. Similarly, we define a schema, `InitNode`, to initialize `Node` by assigning values

to all fields of the node and write a theorem to indicate that there exist a node satisfying the initial conditions of the node.

We define a `Network` as a set of such nodes. The `Network` is composed of all nodes whether they are legitimate (nodes that are still operating in the correct manner and have not been captured) or compromised (nodes that have been captured by an attacker). All nodes that can communicate (operate on a standard frequency) with one another, if they are within the radio range of each other, are part of the `Network`.

$$Network : \mathbb{P}\, Node$$

`CompromisedNode` is a set of nodes and a subset of network. We have not defined legitimate nodes separately as they can be calculated by subtracting the `Network` and `CompromisedNode` sets. We have also not placed any restriction on compromised nodes but later on we will restrict this so that a compromised node cannot be the sink or base station.

$$CompromisedNode : \mathbb{P}\, Node$$

$$CompromisedNode \subseteq Network$$

Wireless sensor network properties are defined in a `WSN` schema. `Neighbour` is defined as a mapping from a node to a set of nodes. Recall that a node may have no neighbour (NULL set) or any number of neighbouring nodes. We model the channel by two message buffers which are updated on reception and transmission. `NodeMessage` and `ChanMessage` are mappings from `Node` to `Message`. A `NodeMessage` is a message recently received by a node and lies inside the node buffer. While `ChanMessage` is the message present in the channel of a node i.e. within the radio range of that node which can be received. As each node has a unique set of neighbours, messages, channels and the domains of all the three must be similar.

WSN

$$Neighbour : Node \rightarrow \mathbb{P}\, Node$$
$$NodeMessage, ChanMessage : Node \rightarrow Message$$

$$\mathrm{dom}\, NodeMessage = \mathrm{dom}\, ChanMessage = \mathrm{dom}\, Neighbour$$

3.3.2 Basic Operations

3.3.2.1 Receive Operation

Now that the specification has defined the messages, nodes and other basics (network, WSN, etc), we define the basic operations that will be used later when defining DoS attacks. A `Receive` operation is defined by the following schema:

Receive

ΔWSN

$Nrec? : Node$

$Mrec? : Message$

$Nrec? \in \text{dom}\, NodeMessage \land Nrec? \in \text{dom}\, ChanMessage \land Nrec? \in \text{dom}\, Neighbour$

$NodeMessage' = NodeMessage \oplus \{(Nrec? \mapsto Mrec?)\}$

$ChanMessage' = ChanMessage$

$Neighbour' = Neighbour$

As this is an abstract model we are not concerned with the internals of the message such as destination, sender etc. In this schema we use the `WSN` schema since its values will be updated. We define two inputs, `Nrec` (the Receive operation will always be performed on a node), and `Mrec` (a message will always be updated). Note that `Nrec` is the receiving node and must be part of the domain of all `WSN` schema. Upon receiving a message, the channel message is unchanged but the node message (or node's buffer for message) is updated as it has received a new message. The `Receive` operation also has no effect on `Node` neighbours. Finally, we define a theorem to confirm that preconditions of `Receive` operation are satisfied.

3.3.2.2 Transmit Operation

In the `Transmit` operation, such as `Receive` we use `WSN` schema as its values will be updated:

Transmit

ΔWSN

$Ntra? : Node$

$Mtra? : Message$

$Ntra? \in \text{dom}\, ChanMessage$

$NodeMessage' = NodeMessage$

$ChanMessage' = ChanMessage \oplus \{(Ntra? \mapsto Mtra?)\}$

$Neighbour' = Neighbour$

Again, the `Transmit` operation will always be performed on a node and a message will always be updated. Therefore, we define both `Ntra` and `Mtra` as inputs . When a node transmits a message, the `NodeMessage` is unchanged as nothing has come from the channel whereas the `ChanMessage` is updated since the node transmits a new message into the channel. The `Transmit` operation has also no effect on `Node` neighbours. A theorem is then used to confirm that the preconditions of the `Transmit` operation are satisfied.

3.3.2.3 In Range Operation

Next, we define a schema which checks whether a node is within the radio range of another node i.e. when a node transmits something, whether or not the other node is able to receive it.

InRange

ΔWSN

$Ncen?, Nran? : Node$

$\forall msg : Message$

$| Ncen? \neq Nran? \land Ncen? \in \text{dom}\, NodeMessage \land Ncen? \in \text{dom}\, ChanMessage$

$\land Nran? \in \text{dom}\, ChanMessage \land Nran? \in \text{dom}\, NodeMessage$

$\bullet\ Transmit[Ncen?/Ntra?, msg/Mtra?] \Rightarrow Receive[Nran?/Nrec?, msg/Mrec?]$

$Neighbour' Nran? = NeighbourNran? \cup \{Ncen?\}$

This schema uses two nodes as inputs: `Ncen`, the current node and `Nran`, the node within the radio range of `Ncen`. Both nodes are in the domain of `NodeMessage` as well as `ChanMessage`, and any message transmitted by `Ncen` is received by `Nran`. These conditions show that `Nran` is within the radio range of `Ncen`. Also, as node `Nran` can listen for messages from node `Ncen`, so `Ncen` is added to the neighbour list of `Nran`. Here `Nran` is not added to the neighbour list of `Ncen` since `Ncen` may not picks up signals of `Nran`. Therefore, `InRange` is a unidirectional link as it checks only that a node is within the radio range of another. A theorem is later employed to check preconditions of this schema.

3.3.2.4 Connected Operation

The Connected schema asserts a bidirectional link between two nodes i.e. in the radio range of each other. It takes as inputs two nodes `Nfir` and `Nsec` and asserts whether they are in the radio range of each other:

Connected

ΞWSN

$Nfir?, Nsec? : Node$

$InRange[Nfir?/Ncen?, Nsec?/Nran?] \land InRange[Nsec?/Ncen?, Nfir?/Nran?]$

3.3.2.5 Neighbour Definitions

We have already seen that a node is within the radio range of another node if `InRange` or the `Connected` operations are used. In practical situation, a neighbour node can be either legitimate or malicious. A Legitimate Neighbour is defined as:

LegitimateNeighbour

ΞWSN

$Ncur?, Ngood? : Node$

$Connected[Ncur?/Nfir?, Ngood?/Nsec?]$

$Ngood? \in Network \land Ngood? \notin CompromisedNode \land Ncur? \in Network$

This schema uses two nodes as input, `Ncur` and `Ngood`. `Ngood` becomes a legitimate neighbour of node `Ncur` because it belongs to the network but does not belong to a set of compromised nodes. A malicious neighbour is defined by:

MaliciousNeighbour

ΞWSN

$Ncur?, Nbad? : Node$

$Connected[Ncur?/Nfir?, Nbad?/Nsec?]$

$(Nbad? \notin Network \lor Nbad? \in CompromisedNode) \land Ncur? \in Network$

`Nbad` is a malicious neighbour of node `Ncur` if either `Nbad` does not belong to the network or belongs to a set of compromised nodes.

3.3.2.6 Cryptography

The cryptography schema describes all the keys or cryptography used in a node. A `PrivateKey` is a unique key assigned to a node whereas the `PairKey` is a key shared by each node within a set of nodes, particularly its communicating neighbours. Note that further details about keys may be added here. We refer to private key as a node key that is not shared with any other node in the network at the the deployment stage. As the specification states a private key is not shared between any of the legitimate nodes. However, the base station or sink might share private keys of all the nodes. Also, for each neighbor of a node `Ncur`, in the network, there is a pair of keys which are shared between the `Ncur` and the neighbour. The cryptography schema is defined as:

Cryptography

$PrivateKey : Node \rightarrowtail KEY$

$PairKey : Node \rightarrowtail \mathbb{P}\, KEY$

ΞWSN

$\forall Ncur, Nother : Node;\ AllNeighbor : \mathbb{P}\, Node \mid Ncur \notin CompromisedNode \land$

$Nother \notin CompromisedNode \land AllNeighbor = NeighborNcur \land$

$Nother.NodeIsSink = FALSE \bullet \forall Nneigh : Node \mid Nneigh \in AllNeighbor$

$\bullet \exists Kpair : KEY \bullet PrivateKeyNcur \neq PrivateKeyNother \land$

$Kpair \in PairKeyNcur \land Kpair \in PairKeyNneigh$

3.3.2.7 Eavesdrop Key

This schema defining the eavesdropping of a shared key, `PairKey`, by an attacker is as:

EavesdropKey

$\Xi Cryptography$

$Ncur? : Node$

$\exists Ngood, Nbad : Node;\ Kpair : KEY$

$\mid LegitimateNeighbour[Ncur?/Ncur?, Ngood/Ngood?]$

$\land MaliciousNeighbour[Ncur?/Ncur?, Nbad/Nbad?]$

$\land Kpair \in PairKeyNcur?$

$\land Kpair \in PairKeyNgood \bullet PairKey' Nbad = PairKeyNbad \cup \{Kpair\}$

Where node `Ncur` is the node being considered and is the input in this schema. The conditions predicate the existence of at least a legitimate neighbour within the radio range of `Ncur` to which `Ncur` was communicating otherwise the data cannot be eavesdropped. There is also a malicious neighbour node, `Nbad`, in the node's radio range from which the attack is launched. `Kpair` is the pair key shared between `Ncur` and `Ngood` for communication and the `EavesdropKey` operation means an attacker's node `Nbad` also gets that key and is added into its pair key list. Note that this operation is also possible when an attacker fabricates keys for a legitimate node without capturing the node `Ncur`. However, this process does not involve capturing the private (individual key) of node which is not exchanged in message passing with neighbours.

3.3.2.8 Node Capture

The capture of a node by an attacker is defined using the following schema:

NodeCapture

ΞWSN

$\Xi Cryptography$

$Ncur? : Node$

$\exists Ngood, Nbad : Node$

$| Ncur?.NodeIsSink = FALSE \land LegitimateNeighbour[Ncur?/Ncur?, Ngood/Ngood?]$

$\land MaliciousNeighbour[Ncur?/Ncur?, Nbad/Nbad?]$

$\bullet PairKey'Nbad = PairKeyNbad \cup PairKeyNcur? \cup \{PrivateKeyNcur?\}$

$\land CompromisedNode = CompromisedNode \cup \{Ncur?\}$

The node `Ncur` is the node being considered and is the input in this schema. The base station is assumed to be secure and cannot be compromised. There is at least one legitimate neighbour within the radio range of `Ncur` to which `Ncur` communicates. There must also be a malicious node, `Nbad`, from which the attack is launched. Once the node is captured, all its keys including the private key are captured by the malicious node or attacker.

3.3.2.9 Encryption Fail

Encryption is considered to have failed in a node if either the current node has been captured or its key has been eavesdropped:

EncryptionFail

ΞWSN

$\Xi Cryptography$

$Ncur? : Node$

$NodeCapture[Ncur?/Ncur?] \lor EavesdropKey[Ncur?/Ncur?]$

3.4 Proposed Formal Specifications for Denial of Service Attacks

Now that the basic operations of WSN have been defined in section 3.3, let us define DoS attacks using Z specifications:

3.4.1 Wormhole Attack

In a wormhole attack, an attacker records packets (or bits) at one location in the network, tunnels them to another location, and retransmits these into the network. A wormhole attack not only hinders correct routing but is also the precursor to many other attacks such as black hole, sink hole, etc. In a wormhole attack the tunnel is usually a low latency link. However,

wormhole is tunnel spanned over multiple hops and we assume that the delay in propagation depends on the number of hops a message has traveled, thus making the wormhole tunnel a low latency link. Perhaps, the specifications can be improved by defining a `HopCount` metric and by considering that the two legitimate nodes at the end of the tunnel are more that four hops away (HopCount>4). This may be the subject of future work. Our definition of a wormhole attacker is described in the following schema:

$$
\begin{array}{|l}
\textit{WormholeAttack} \\
\hline
\Xi\, WSN \\
Nleg1?, Nleg2? : Node \\
\hline
\exists\, Nmal1, Nmal2 : Node \\
\mid Connected[Nmal1/Nfir?, Nmal2/Nsec?] \land \neg\, Connected[Nleg1?/Nfir?, Nleg2?/Nsec?] \\
\land\, MaliciousNeighbour[Nleg1?/Ncur?, Nmal1/Nbad?] \\
\land\, MaliciousNeighbour[Nleg2?/Ncur?, Nmal2/Nbad?] \bullet \forall\, m1, m2 : Message \\
\bullet\, (Transmit[Nleg1?/Ntra?, m1/Mtra?] \Rightarrow Receive[Nleg2?/Nrec?, m1/Mrec?] \\
\land \neg\, Receive[Nleg1?/Nrec?, m1/Mrec?]) \land (Transmit[Nleg2?/Ntra?, m2/Mtra?] \\
\Rightarrow Receive[Nleg1?/Nrec?, m2/Mrec?] \land \neg\, Receive[Nleg2?/Nrec?, m2/Mrec?]) \\
\hline
\end{array}
$$

In this type of attack, a tunnel (can be wired or wireless) exists between the two malicious nodes `Nmal1` and `Nmal2`. Nmal1 is the malicious hidden neighbour in the legitimate node `Nleg1`'s radio range and `Nmal2` is the malicious neighbour of `Nleg2`. As the two legitimate nodes `Nleg1` and `Nleg2` are not connected i.e. there is no link between the two, they must not hear each other. Due to the wormhole tunnel, whatever the two nodes transmit is received by the other. Thus a virtual link exists between the two legitimate nodes via the wormhole tunnel. Note that both legitimate nodes never receive back their own message indicating that the attacker used a hidden channel for tunnelling.

3.4.2 Invisible Node Attack (INA)

Researchers define the wormhole attack in two different ways. One definition uses the out of bound channel as defined in Section 3.4.1 where two attacker nodes are required. Some argue that the wormhole is even possible with a single attacker's node using the packet relay. But the packet relay is also defined as an invisible node attack (INA) by many researchers and we also treat it as a different type of attack from wormhole attack in which, the attackers range is different, more devastating and the use of hidden channel means the attackers remain undetected. On the other hand, in INA the radio range is limited by the attacker's capability and because it uses the same channel and the nodes also detect the same message coming back to them. Note that in the specifications of INA, the same attacker is a neighbour of both the legitimate nodes. This makes the specifications of INA quite similar to a wormhole attack in

a sense that there is one malicious node which is a neighbour of both unconnected legitimate nodes and the legitimate nodes can receive the same message back when an attack is launched.

InvisibleNodeAttack

$$\Xi WSN$$
$$Nleg1?, Nleg2? : Node$$

$$\exists Nmal : Node \mid \neg\ Connected[Nleg1?/Nfir?, Nleg2?/Nsec?]$$
$$\land\ MaliciousNeighbour[Nleg1?/Ncur?, Nmal/Nbad?]$$
$$\land\ MaliciousNeighbour[Nleg2?/Ncur?, Nmal/Nbad?] \bullet \forall m1, m2 : Message$$
$$\bullet\ (Transmit[Nleg1?/Ntra?, m1/Mtra?] \Rightarrow Receive[Nleg2?/Nrec?, m1/Mrec?]$$
$$\land\ Receive[Nleg1?/Nrec?, m1/Mrec?]) \land (Transmit[Nleg2?/Ntra?, m2/Mtra?]$$
$$\Rightarrow Receive[Nleg1?/Nrec?, m2/Mrec?] \land Receive[Nleg2?/Nrec?, m2/Mrec?])$$

3.4.3 Black hole Attack

In a black hole attack, a malicious node joins the network and then either discards all the messages it receives or performs selective forwarding. Our specifications for this attack is as follows:

BlackholeAttack

$$\Xi Cryptography$$
$$Ncur?, Nbh? : Node$$

$$\exists Nc : Node \mid Nc \in Network \land Nbh?.NodeIsSink = FALSE$$
$$\land\ Connected[Ncur?/Nfir?, Nbh?/Nsec?] \land Connected[Nbh?/Nfir?, Nc/Nsec?]$$
$$\land\ (WormholeAttack[Ncur?/Nleg1?, Nbh?/Nleg2?] \lor NodeCapture[Nbh?/Ncur?])$$
$$\lor\ InvisibleNodeAttack[Ncur?/Nleg1?, Nbh?/Nleg2?]$$
$$\bullet \exists m1, m2 : Message \mid m1.MsgData = m2.MsgData$$
$$\bullet\ Transmit[Ncur?/Ntra?, m1/Mtra?] \Rightarrow Receive[Nbh?/Nrec?, m1/Mrec?]$$
$$\land \neg\ Transmit[Nbh?/Ntra?, m2/Mtra?]$$

`Ncur` and `Nbh` are the input nodes with `Ncur` being the current node from which the attack is launched while `Nbh` is the node that acts as the black hole neighbour. `Nc` is another node that is part of the network and is connected to the `Nbh` node to which `Nbh` can forward the data. Also `Nbh` is not a sink so node `Nbh` must forward the data it receives in normal conditions. `Nbh` becomes a black hole because either this node is captured or there is a hidden attacker (wormhole tunnel/INA) between `Nbh` and `Ncur`. We have already defined encryption failure as a result of `Nbh` having been captured by the attacker or its keys captured via other means. In order to perform a black hole attack node `Nbh` upon receiving data from `Ncur`, does not

retransmit it. It is worth noting that the messages `m1` and `m2` have the same data but different sender and destination IDs.

3.4.4 Spoofing Attack

A spoofing attack is the process of altering or replaying routing information (data, beacon or acknowledgement packets). It can lead to the creation of inaccurate or unstable routes. A spoofing attack can be defined as (i) direct spoofing, in which an attacker node, upon receiving a message, changes the contents of the message before retransmitting it or (ii) indirect spoofing, in which an attacker, node upon eavesdropping a message, sends it to another node by modifying the message. Indirect spoofing is defined as:

IndirectSpoofing

$$\Xi Cryptography$$
$$Ncur?, Natk? : Node$$

$$\exists Ngood : Node \mid Ngood \in Network \land Connected[Ncur?/Nfir?, Natk?/Nsec?]$$
$$\land Connected[Ncur?/Nfir?, Ngood/Nsec?] \land EncryptionFail[Natk?/Ncur?]$$
$$\bullet \exists m1 : Message \mid m1.MsgSender = Ncur?.NodeID \land$$
$$m1.MsgDestination = Ngood.NodeID \bullet \exists d : \mathbb{N} \mid m1.MsgData = d$$
$$\bullet Transmit[Ncur?/Ntra?, m1/Mtra?] \Rightarrow Receive[Natk?/Nrec?, m1/Mrec?]$$
$$\Rightarrow (\exists m2 : Message \mid m2.MsgSender \neq m2.MsgSender \lor$$
$$m2.MsgDestination \neq m1.MsgDestination \lor m1.MsgData \neq d$$
$$\bullet Transmit[Natk?/Ntra?, m2/Mtra?])$$

This schema has two inputs the current node, `Ncur`, and an attacker neighbour node, `Natk`. There exists a legitimate node, `Ngood`, connected to `Ncur` to which `Ncur` transmits the message. As `Natk` is a neighbour of `Ncur` it also receives this message and modifies some of the contents of the message. It then retransmits this spoofed message into the channel. Note that the contents of `m2` are not correct and an attacker has either modified parts(s) of or the whole message. Direct spoofing is defined as:

DirectSpoofing

$$\Xi Cryptography$$
$$Ncur?, Natk? : Node$$

$$Connected[Ncur?/Nfir?, Natk?/Nsec?] \land EncryptionFail[Natk?/Ncur?]$$
$$\exists m1 : Message \mid m1.MsgSender = Ncur?.NodeID \land m1.MsgDestination = Natk?.NodeID$$
$$\bullet \exists d : \mathbb{N} \mid m1.MsgData = d \bullet Transmit[Ncur?/Ntra?, m1/Mtra?]$$
$$\Rightarrow Receive[Natk?/Nrec?, m1/Mrec?] \Rightarrow (\exists m2 : Message \mid m2.MsgSender \neq$$
$$Natk?.NodeID \lor m2.MsgData \neq d \bullet Transmit[Natk?/Ntra?, m2/Mtra?])$$

In a direct spoofing attack the attacker, upon receiving a message, either retransmits it with a fake sender or alters its data. Note that the destination is not important here rather, the node ID to which the message is forwarded is more important. The spoofing attack is of either direct spoofing or indirect spoofing:

SpoofingAttack

$\Xi Cryptography$

$Ncur?, Natk? : Node$

$DirectSpoofing \lor IndirectSpoofing$

3.4.5 False-Injection Attack

A false injection attack refers to the introduction of extra data or control packets into the network. It consumes bandwidth and may cause routing loops. The aim of this attack is to consume resources wastefully. In our definition of a false injection attack a malicious neighbor `Natk`, of legitimate node `Ncur`, injects extra message packets into a network. Unlike the spoofing attack, the message injected might not be the same as the one received earlier (`Mr`) or some fake message, `Mf`. Therefore, the main aim here is to add extra traffic. It is also different from INA as the attacker may inject the messages many times.

FalseInjectionAttack

$\Xi Cryptography$

$Ncur?, Natk? : Node$

$MaliciousNeighbor[Ncur?/Ncur?, Natk?/Nbad?]$

$\exists Mr, Mf : Message \mid Receive[Natk?/Nrec?, Mr/Mrec?]$

- $\exists FalseMsgs : \mathbb{P}\, Message$
- $\forall Mt : Message \mid Mt \in FalseMsgs \land (Mt = Mr \lor Mt = Mf)$
- $Transmit[Natk?/Ntra?, Mt/Mtra?]$

3.4.6 Sybil Attack

A sybil attack occurs when a malicious node presents multiple identities simultaneously within the network. Such a node may be used to subvert the routing protocols that rely on redundancy, such as multi-path protocols. The specifications of a sybil attack states that a malicious neighbour, `Natk` uses a set of fabricated IDs, `FalseIds`, instead of using its original node ID as message sender in the message transmission. The fabricated IDs must be an ID of a node present in the Network. How the attacker gets these IDs is not important here.

SybilAttack

$\Xi Cryptography$

$Ncur?, Natk? : Node$

$MaliciousNeighbor[Ncur?/Ncur?, Natk?/Nbad?]$

$\forall m : Message;\ FalseIds : \mathbb{P}\,\mathbb{N} \mid m.MsgSender \in FalseIds \land Natk?.NodeID \notin FalseIds$

$\bullet\ \exists N : Node \mid N.NodeID \in FalseIds \land N \in Network$

$\bullet\ Transmit[Natk?/Ntra?, m/Mtra?]$

3.4.7 Node Replication Attack

A node replication attack is similar to a sybil attack except that the attacker uses the same identity (`SameId`) for multiple nodes. Thus, a single adversary node may represent many virtual locations in the network. These multiple nodes must be `MaliciousNodes` because these are either the attackers or compromised nodes. All these malicious nodes have a common ID i.e. `SameId`. One of these malicious nodes, `Natk`, lies within the radio range of the current node `Ncur`. Thus whenever `Natk` transmits a message, the sender ID is always the ID, `SameId`.

NodeReplicationAttack

$\Xi Cryptography$

$Ncur?, Natk? : Node$

$\exists MaliciousNodes : \mathbb{P}\,Node;\ SameId : \mathbb{N};\ m : Message \mid \forall Badnode : Node$

$\bullet\ Badnode \in MaliciousNodes \land Badnode.NodeID = SameId \land (Badnode \notin Network$

$\lor\ Badnode \in CompromisedNode) \land m.MsgSender = SameId \land Natk? \in MaliciousNodes$

$\land\ Connected[Ncur?/Nfir?, Natk?/Nsec?] \bullet Transmit[Natk?/Ntra?, m/Mtra?]$

3.4.8 Hello-Flood Attack

A hello flood attack involves the use of a high power transmitter by an attacker to broadcast routing or other information, with the purpose of convincing every node within the radio range that the attacker is a legitimate neighbour. The attacker may then be established in routes that are unusable by other nodes since their transmitters are much less powerful. The specifications state that attacker node `Natk` has a number of nodes within its radio range from the `Network` described as `allnodes`. Note that these links are unidirectional (`InRange`), e.g. `Natk` can be a laptop class or a powerful transmission node i.e. can broadcast the message with large transmission power. Only some of the nodes, `somenodes`, have this attacker node within its radio range as well. The remaining nodes do not have `Natk` in their radio range and the current node `Ncur`, is part of that group. Therefore, whenever an attacker node transmits, it is heard by all nodes and thus could convince most nodes in the network that `Natk` is their neighbour.

HelloFloodAttack

$\Xi Cryptography$

$Ncur?, Natk? : Node$

$\exists\, m : Message;\ allnodes, somenodes : \mathbb{P}\, Node$

$\mid somenodes \subseteq allnodes \subset Network \land somenodes \neq \emptyset \land Ncur? \in allnodes \setminus somenodes$

$\bullet\ \forall\, Nall, Nsome, Nrem : Node \mid Nall \in allnodes \land Nsome \in somenodes$

$\land\ Nrem \in allnodes \setminus somenodes \bullet InRange[Natk?/Ncen?, Nall/Nran?]$

$\land\ InRange[Nsome/Ncen?, Natk?/Nran?] \land \neg\ InRange[Nrem/Ncen?, Natk?/Nran?]$

$\land\ Transmit[Natk?/Ntra?, m/Mtra?]$

3.4.9 Jamming Attack

Jamming is a physical layer attack instigated by creating radio noise in a particular physical area. The specifications for a jamming attack state that there is a malicious neighbour, `Natk`, within its radio range. When `Ncur` transmits a messages some of its neighbours (`someneighbors`) receive this transmission but the remaining neighbouring nodes do not receive due to the RF noise created by `Natk`. Note that the receiving neighbours might be null as stated by the specifications. Thus, jamming can prevent a few or all neighbours to receive the messages transmitted by the node `Ncur`.

JammingAttack

$\Xi Cryptography$

$Ncur?, Natk? : Node$

$MaliciousNeighbor[Ncur?/Ncur?, Natk?/Nbad?]$

$\forall\, m : Message;\ allneighbors, someneighbors : \mathbb{P}\, Node \mid someneighbors \subseteq allneighbors$

$\subset Network \land someneighbors \neq \emptyset \land allneighbors = NeighborNcur?$

$\bullet\ \forall\, Nall, Nsome, Nrem : Node$

$\mid Nall \in allneighbors \land Nsome \in someneighbors$

$\land\ Nrem \in allneighbors \setminus someneighbors \bullet Connected[Ncur?/Nfir?, Nall/Nsec?]$

$\land\ Transmit[Ncur?/Ntra?, m/Mtra?] \Rightarrow Receive[Nsome/Nrec?, m/Mrec?]$

$\land\ \neg\ Receive[Nrem/Nrec?, m/Mrec?]$

3.4.10 Sinkhole Attack

The specifications for this attack are:

$$
\begin{array}{|l}
\textit{SinkholeAttack} \\
\hline
\Xi \textit{Cryptography} \\
\textit{Ncur?}, \textit{Natk?} : \textit{Node} \\
\hline
\textit{HelloFloodAttack}[\textit{Ncur?}/\textit{Ncur?}, \textit{Natk?}/\textit{Natk?}] \\
\lor \textit{SpoofingAttack}[\textit{Ncur?}/\textit{Ncur?}, \textit{Natk?}/\textit{Natk?}] \\
\lor \textit{WormholeAttack}[\textit{Ncur?}/\textit{Nleg}1?, \textit{Natk?}/\textit{Nleg}2?] \\
\lor \textit{InvisibleNodeAttack}[\textit{Ncur?}/\textit{Nleg}1?, \textit{Natk?}/\textit{Nleg}2?] \\
\Rightarrow \textit{Network} = \textit{Network} \cup \{\textit{Natk?}\} \land \textit{BlackholeAttack}[\textit{Ncur?}/\textit{Ncur?}, \textit{Natk?}/\textit{Nbh?}] \\
\hline
\end{array}
$$

In a sinkhole attack, a malicious node, `Natk`, attracts all the surrounding traffic by making itself attractive to all nodes within the radio range. This attack is possible if `Natk` with a long range can provide nodes within the radio range a smaller hop path (hello flood), claim that `Natk` is the base station itself (or near to the base station) by a spoofing attack, or provides a shorter/faster hop path using wormhole/INA. After adding itself to the network by becoming attractive the attacker launches a black hole attack and thus drops data packets.

3.5 Chapter Summary

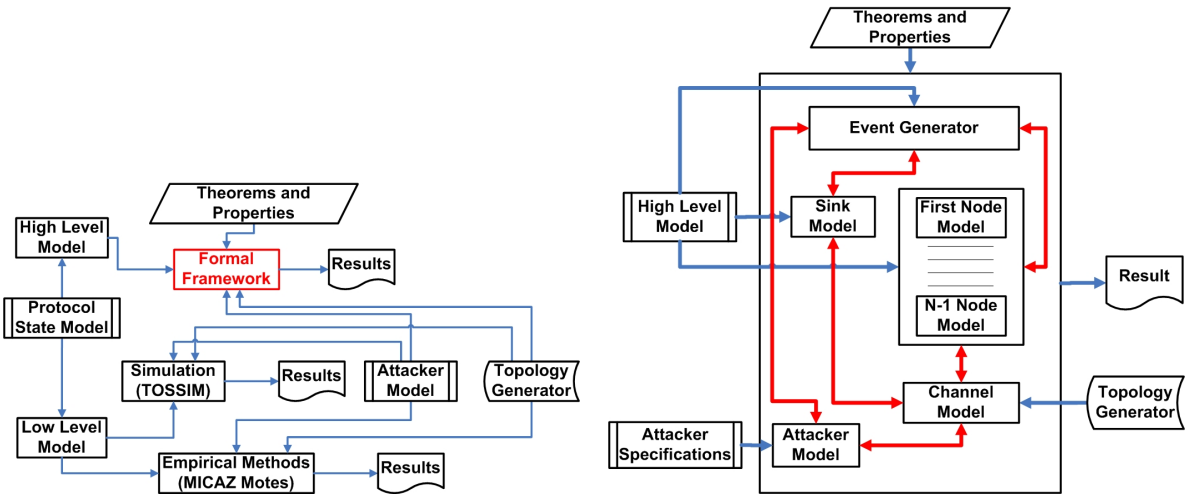
This chapter formally defines WSNs and DoS attacks which are considered in this research. The aim of these specifications is (i) to define the attacks separately as one generalized attacker model does not have sufficient details and (ii) to present specifications of different DoS attacks in a precise and formal manner. The specifications are represented from grass root level so Z is used as a formal mathematical notation useful for detailed description. The use of accepted programming/modelling techniques such as Z provides advantages that researchers agree on the same specification and the specification, which should be readable and acceptable, so that a solutions to DoS attacks can be found on the same sound basis. It also gives the research a possibility to annotate the DoS specification in a structured way thus enabling future modification. Many published and widely recognized WSN attacks are formally defined, a task which has not been done before. Researchers have used different definitions for same attacks and there exists no concise and formal definition so far. This research attempts to fill this gap by writing the specifications of DoS attacks in a formal way. Once the specifications of the attacks were made using Z notation; this work can assist in the development of a formal framework used to check different routing protocols. This can be seen as the foundation where specifications can be improved in the future. Moreover, these specifications present just an abstract model for different attacks and more detailed models can be extracted in conjunction with the specifications of a routing protocol thereby assisting the researchers in detecting the vulnerabilities of the protocols against different DoS attacks.

Chapter 4

Formal Analysis of Routing Protocols

4.1 Introduction

This dissertation presents an innovative design method (a combination of formal model-checking and simulation) to address denial of services (DoS) attacks on WSN routing. Formal modelling is applied to detect the attacks in some published routing protocols, the results of which are supported by computer simulation. It also confirms the results achieved by formal modelling are repeatable. Many DoS attacks have been recognized in the literature [15, 13]. The vulnerability of routing protocols to these attacks has been discussed and new protocols have been developed or older protocols modified to guard against them [15]. But the techniques used so far generally rely on visual inspection and simulation and often are not adequate for the detection of worst case scenarios. The work described in this chapter investigates to what extent the application of formal methods leads to more effective bug detection in routing protocols for WSNs. Finite state models of protocols are described and simple specification properties are checked. Properties are checked for all possible topologies of N nodes, where N is typically small (< 6) in order to allow automatic verification to complete with the available computing resources (memory and CPU time). This has proved to be an effective bug-hunting approach and weaknesses in existing protocols have been discovered. The work presented here extends and improves the methodology adopted by Andel [227]. This chapter is organized as follows: Section 4.2 explains the methods adopted to analyse previous protocols and to evaluate rigorously a newly developed protocol for use against DoS attacks. A formal framework is presented in Section 4.3 which is used to the resilience of wireless routing protocols against DoS attacks. An example of a routing protocol, flooding, is also presented in Section 4.4 to illustrate the application of the formal framework. The application of the framework to different routing protocols is explained in Sections 4.5 to 4.10. Finally, a summary of the chapter is presented in Section 4.11.



(a) Methods adopted main block diagram

(b) Formal Framework Block Diagram

Figure 4.1: Approach adopted in the research

4.2 Methods Adopted in the Research

This research addresses DoS attacks on WSN routing. Formal modelling (Section 2.6), is applied to detect the attacks in published routing protocols and a newly developed protocol. The results achieved from formal evaluation are supported by computer simulation. By using formal methods, not only are the assumptions of protocols clearly modelled, but also all possible behaviours of the models can be examined for error conditions. This enables strong guarantees on behaviour for a range of scenarios, including scenarios in which nodes can fail. Formal model checking has two main advantages over other techniques (simulation and testing):

1. There is no need to build a prototype of the system,

2. It is possible to check all execution paths of the system. This is very important because simulation or testing is likely to examine only a small fraction of possible states and cannot provide assurance that the whole system behaves as expected (some errors may remain hidden until the system is in the production stage).

The methods adopted to analyse wireless routing protocols is shown in Figure 4.1(a). The *protocol state model* is a semi formal definition of a routing protocol. The semi formal definition includes message sequence diagrams and message passing expressed as formal equations. Both high level and low level models are then generated from the state model of a routing protocol. The *high level model* was converted into a formal model and specification properties defined to check the presence of any faults (vulnerability to attacks) present in the routing protocol. The properties included basic *sanity* checks (confirmation that the model possesses some fundamental properties, debugging checks, etc.), the *liveness* (something good will eventually occur) and the *safety* (nothing bad ever occurs). In case a property fails, the formal model-checker automatically generates a trace providing the reason as to how the attack occurred in the protocol.

The *low level model* is then used to implement the protocol in a low level simulator (and perhaps a practical implementation on hardware). The same topology were used to confirm that the results of the formal model could be replicated by simulation. The topologies are mostly fed manually one at a time and are the model is then checked against properties. This is done to save state space. For certain cases, when the model is simple an automatic topology generator is used inside the model.

The simulator used is TOSSIM which operates at the bit level (high fidelity). Moreover, the TOSSIM simulation code can directly be programmed on hardware without any modification. The practical implementation is only required if one wants to check the real world radio model. Environmental effects are not the main concern in this research so the practical implementation is used sparingly.

This research deliberately considered only routing protocols whose semi-formal definitions are available in the literature. Thus, in order to check a routing protocol using this framework, the only requirement is to write the protocol specifications in a semi-formal notion or as message sequence diagrams. The drawback however in the framework is that the process is not automatic and a formal model has to be developed manually.

4.3 Formal Framework

In order to rigorously check routing protocols against DoS attacks, a modelling framework has been developed that supports an automatic analysis of the protocols. Models were initially developed and analysed using PROMELA and Spin [47, 196]. But it was discovered that more compact and efficient models can be developed in Uppaal [197] and Spin/PROMELA was abandoned.

4.3.1 The Framework

The framework comprises a model for the wireless medium and models representing wireless nodes acting in one of the following roles: source, sink (BS), normal node or attacker. Each wireless node model is instantiated from a Uppaal template that represents the node behaviour as determined by the role in the protocol and the attack under consideration. The block diagram of the formal framework is shown in Figure 4.1(b). The formal model comprises 5 main parts: attacker model, sink model, channel model, event generator (EG) model and node models. The protocol is checked against different DoS attacks independently, thus the *attacker model* is replaced for each specific attack. Apart from INA and wormhole attackers, which are modelled separately (the attacking node(s) are hidden and forward all messages except data), the remaining attacks are incorporated inside the node model in a function `GenerateAttack()`. A *black hole* is modelled simply by modifying the node model which forwards all messages correctly except data messages. A *hello flood* attack is modelled by modifying the connection

matrix to represent the fact that the attacker can establish unidirectional link with all the other nodes in the network. A *sinkhole* attacker is modelled by having the attacker, initiate the broadcast of a BS. Finally, the spoofing attacker is a node model which transmits the messages by replacing the ID of the source node instead of its own.

The *channel model* represents the topology and the capacity for communication between both legitimate (including BS) and malicious nodes. The channel model is explained in detail later in this section. The remaining three models are developed using the high level model of a particular routing protocol.

The *node* model is developed by transforming the semi formal information, in the protocol state model, into a formal way. The node model contains a number of states depending on the protocol's specifications. Each particular message passed between nodes in a protocol enables at least 2 states in the node model 'send' and 'receive'. Sometimes more than 2 states are needed, e.g. before sending the data from the source node a 'sense' state models sensing data from environment. Apart from these states there is always a state in which a node does nothing and remains idle (listen state). Some other states in the node model are the 'finish' and 'initial' states, indicating the starting and the terminating states of a protocol.

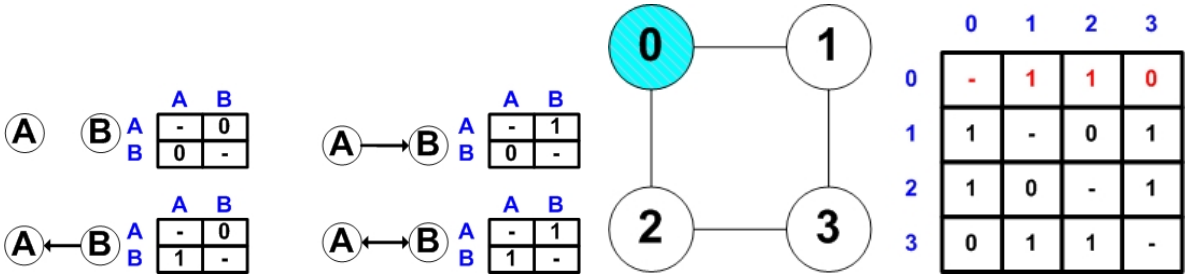
Multiple concurrent node models are used in the formal framework because WSN comprised of more than one node. The node can be a source, a target, the destination or relay (intermediate) depending upon particular routing protocol requirements. The *base station (BS)* or *sink* is also a node. But in order to save the state space, sometimes, the framework models it separately from the node model. The reason for this is that the sink model does not employ complex functionality as required in nodes and some details can be removed in modelling. When an N node network is employed with one BS, there are N-1 node models and one sink model.

Finally the *event generator model* is used to generate different events required in the protocol. The events are triggers to enable nodes to sense data from environment, generating a timer's timeout or finishing a particular phase, etc. In Uppaal, sometimes, instead of using quantified time (clocks), an event generator is employed to generate timeouts to indicate that a phase has been completed.

4.3.2 Modelling Topologies

The essence of our approach is to model N nodes and to check a network's resistance against DoS attacks for all possible network topologies. A network topology is represented by a boolean $N \times N$ matrix, where N is the total number of nodes in the network. A '1' in the matrix at (i, j) indicates the presence of an RF link between nodes i and j; '0' indicates the absence of an RF link between nodes i and j.

This approach has been used by other researchers [214, 228, 37] to model network topologies. For a 2 node network all possible topologies and the matrix values are shown in Figure 4.2(a).



(a) All possible 2 node topologies

(b) Topology matrix for a 4 node network

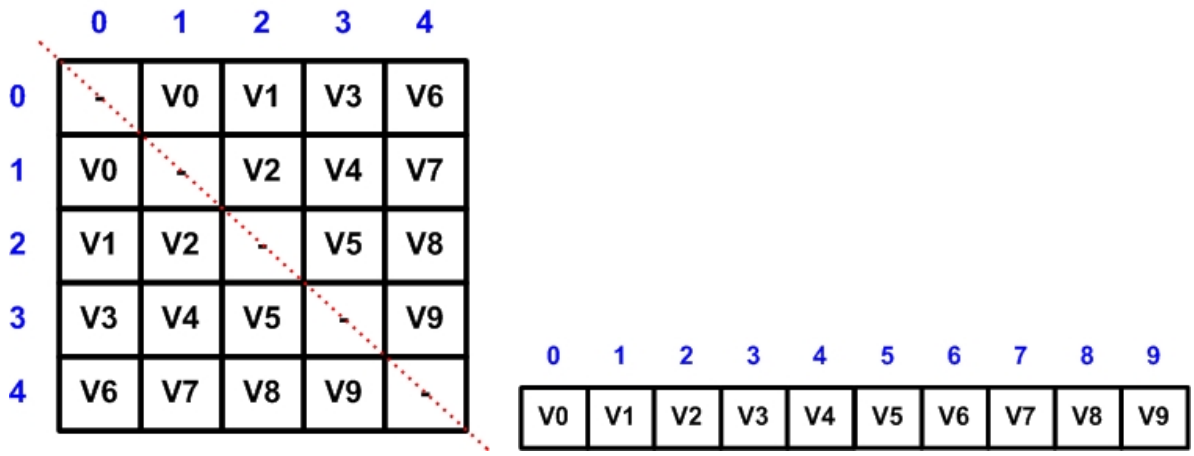
Figure 4.2: Examples of Connectivity Matrix

The arrow head in the figure indicates that message can be received by that node. The four possible cases are: (i) A and B are not connected (ii) B can hear A (iii) A can hear B and (iv) Both A and B can hear one another. A larger 4 node network is shown in Figure 4.2(b) for a single topology. All the links are bidirectional so the arrow notation has been dropped. The framework checks all the possible topologies for a given number of nodes N.

Table 4.1: The number of symmetric and asymmetric topologies for a given number of nodes

Nodes	**Number of topologies for Symmetric links**	**Number of topologies for Asymmetric links**
N	$2^{\frac{N(N-1)}{2}}$	$2^{N(N-1)}$
2	2	4
3	8	64
4	64	4096
5	1,024	1,049,000
6	32,768	1,074,000,000

The number of possible links is shown by Table 4.1 for different node networks. The number of topologies increase quadratically if the links are asymmetric instead of symmetric. So symmetric links are usually assumed. For asymmetric links, networks of up to 4 nodes were checked. Note that some protocols, including the new protocol developed in this research, remove unidirectional links before routing data and thus there is no need to check for asymmetric cases. Moreover, checking all the possible combinations for with symmetric links is not necessary. This is because in some of these networks the source is not connected to the sink. As the research considers only attacks that prevent data from reaching the sink these topologies are



(a) Matrix converted from array for 5 node network with symmetric links

(b) Array used in a 5 node network

Figure 4.3: Connectivity Matrix Implementation

omitted. In symmetric networks the matrix can be represented simply by an array as shown in Figure 4.3(b).

4.3.3 Assumptions for the Formal Framework

For the formal framework it is assumed that:

- The channel is error-free, i.e. no message is lost in the communication. The presence of bugs detected in such an ideal radio environment is a clear indication that a protocol will exhibit similar problems in a more realistic RF environment. When required, a noise model is introduced in the model that enables a network to lose some messages non-deterministically.
- The channel is collision free. It is assumed that the link layer protocol eliminates loss resulting from collisions.
- Networks have symmetric links unless stated otherwise.
- Network contains at least one legitimate path (without an attacker) between the source and sink nodes.

4.3.4 Modelling a Channel

Broadcast message passing in Uppaal conveniently models WSN message passing. The channel is modelled using a global flag `ChannelBusy` to indicate whether a channel is busy or free. This flag can be used to check if a node is allowed to broadcast a message or not. Multiple receptions

are modelled by another variable, `BusyNodes`. The node models wait, where appropriate, until this variable becomes 0, signalling that all the recipients have performed their necessary actions and are now free. Message reception is performed by checking the `Topology` matrix. In case the message is unicast instead of broadcast, the guard contains additional information so that only the message addressed to a node is received. The rest of the nodes ignore this message. The contents of the message are declared as global variables starting with `Msg_`. Once a node needs to transmit any message it sets the `ChannelBusy` flag and then updates these global variables. The receiving nodes increment `BusyNodes` and read these global variables. No further transmission is allowed unless the `ChannelBusy` flag is cleared and `BusyNodes` becomes 0. Unicast messages have an additional guard for `Msg_ID`. However, in most cases, another global variable `Node` is used to indicate the actual message sender in the `Topology` matrix. This is because the `Msg_ID` is a field of a message and is sometimes used as a sender and destination ID for some protocols (Section 4.5).

4.4 The Flooding Protocol

To show the application of the formal framework and the methods adopted in this thesis an example of the Flooding protocol is presented. In Flooding, each node, on receiving any data packet, broadcasts it to all of its neighbours. This process is repeated until the packet arrives at the sink or destination or the maximum number of hops for the packet is reached. Duplicate message packets are suppressed to avoid duplicate messages sent by the same node.

4.4.1 Semi Formal Notation

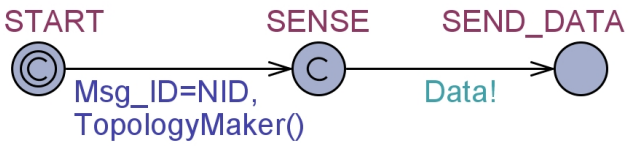
The first step is to express this informal definition of the protocol as a series of formal communications (equations). These are then later transformed into a formal model. There is only one message exchanged between the nodes i.e. the broadcast of data. The semi formal notation for this message is:

$$N \rightarrow * : (data, ID_N)$$

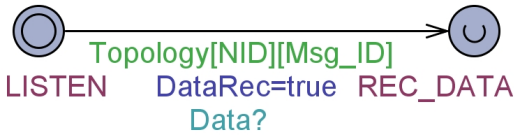
The notation $A \rightarrow B : M$ is used in this thesis to denote that the node A transmits the message M to the node B. A broadcast transmission is indicated by the use of $*$ in place of B indicating that message M is transmitted to all nodes within radio range of A.

4.4.2 Formal Model

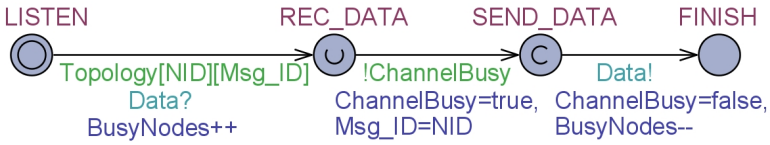
As indicated earlier, a formal model in the framework comprises of (i) a send and a receive location indicating each message exchanged between nodes in the protocol (ii) additional functions in a protocol e.g. sense environment location (iii) a start and finish location and (iv) a listen location in which a node continuously monitors the channel to receive a message. These



(a) Source model



(b) Sink model



(c) Node model

Figure 4.4: UPPAAL model for Flooding protocol

are evident by looking at the model parts which comprises a source node, a sink node and a node model as shown in Figure 4.4(a) 4.4(b) and 4.4(c) respectively. As only data messages are exchanged between nodes, the two respective locations are SEND_DATA and REC_DATA.

The *source model* acts as source node, this simply senses data (SENSE) and transmits it (SEND_DATA). In later models, an event generator (EG) model is used to trigger a node to sense data in this case the EG model is not required. The first location (initial) of the source model START is critical. This means this location is restricted and among all the active states the only possible transition is from this location. Node topologies are generated on taking this transition using the `TopologyMaker()` function, which simply converts the array (presented in Figure 4.3(b)) into the `Topology` matrix (presented in Figure 4.3(a)). The next location in the source model (SENSE) is also critical immediately enabling the next transition.

The source node transmits the data as indicated by 'Data!' and the receiver nodes receive

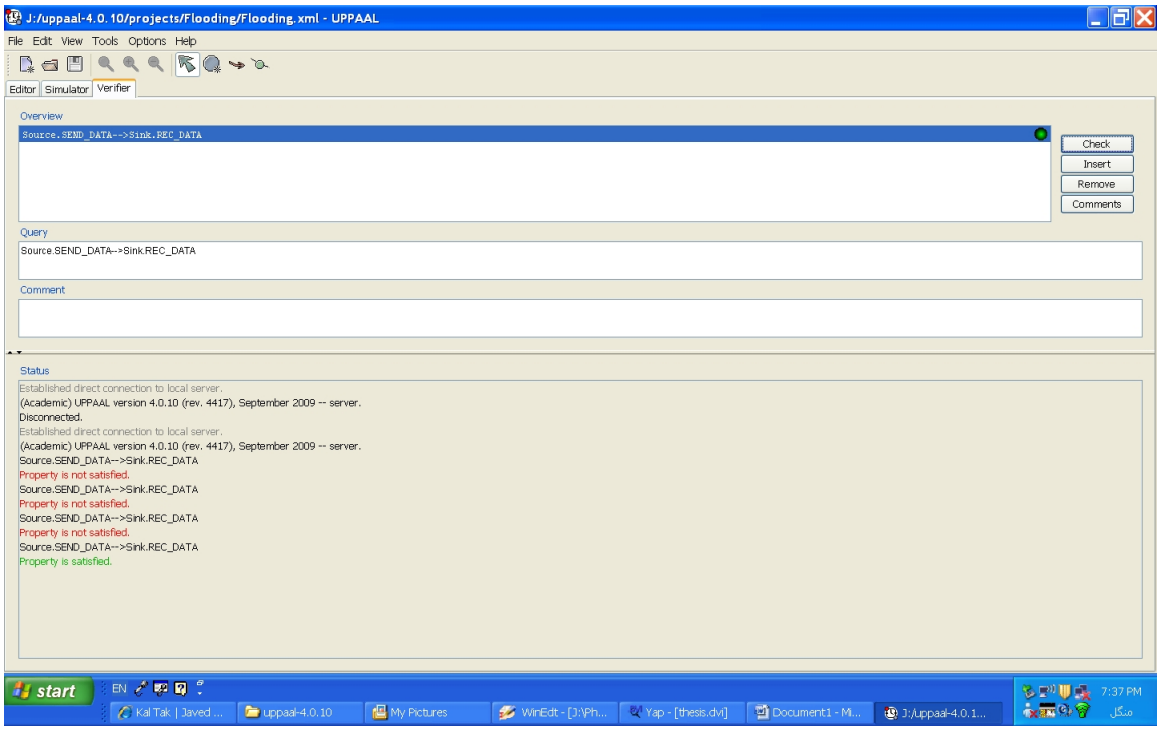
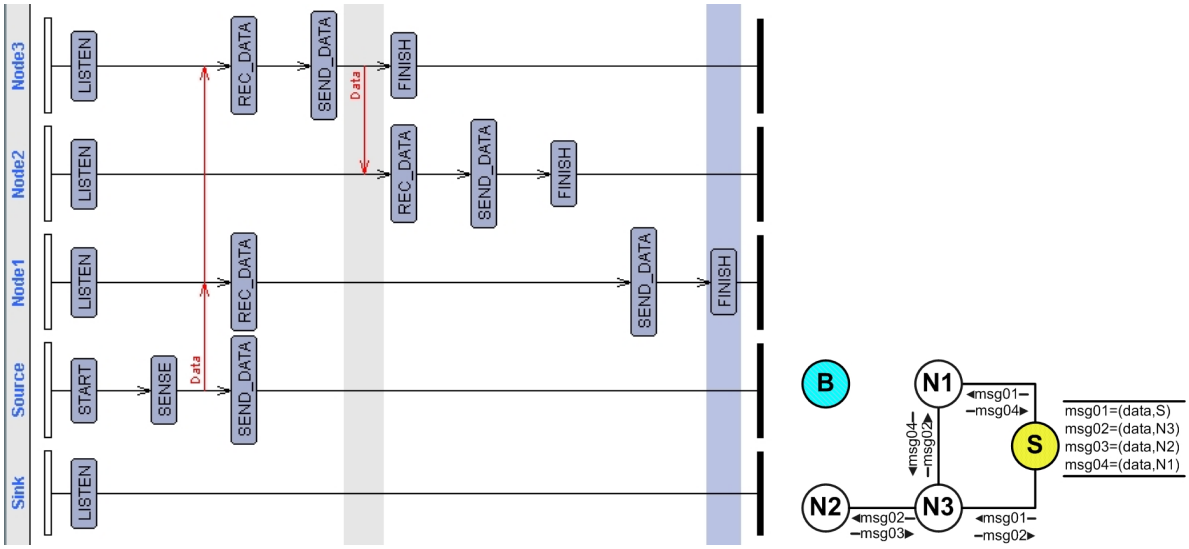


Figure 4.5: Confirmation by Uppaal that a topology satisfies the property

it using 'Data?'. As the message is broadcast all node models will receive this message. Each node model utilizes the `Topology` matrix in a guard to determine if it is within the transmiting node's radio range (can receive the message). The receiver node increments the `BusyNode` variable to indicate there is at least one node busy in some receiving operation. This variable is decremented when the receiver node becomes free again (FINISH location). The receiver node then continuously monitors the `ChannelBusy` flag and can only transmits once this flag is false (guard before the REC_DATA location). Once the flag is found to be false, the node enters the transmission phase; the node sets the `ChannelBusy` flag and transmits immediately. Note that the SEND_DATA location is critical indicating this state must be executed next time at all costs. Once data is transmitted (Data!), the `ChannelBusy` flag becomes false again allowing any other node to transmit. Finally, the global variable `Msg_ID`, which indicates the sender ID in the flooding protocol, is assigned the node's current ID (`NID`) before message transmission. The value of this global variable is updated only after a node makes the `ChannelBusy` flag true. This prevents access to this global variable by any other node at that time. Similarly in all the other protocols examined in this thesis, all message fields (starting with `Msg_`) are updated by making `ChannelBusy` true. The implemented flooding protocol will also confirm if the source and the sink are connected in a given topology. If both are connected the data message from the source will eventually reach the sink.

The *node model* represents all the intermediate nodes. A node stays in the LISTEN location until it receives a data message (REC_DATA). It then waits for the channel to become free and



(a) Original Uppaal Trace

(b) Trace presented in this Thesis

Figure 4.6: Trace generated by Uppaal in case the property fails

retransmits the data (SEND_DATA). When the channel becomes free, it moves to the FINISH location and deadlocks there. A *sink model* remains in the LISTEN location until it receives the data and then moves to the REC_DATA location.

4.4.3 Verification

The verification process involves first confirming some *sanity* checks e.g. whenever a source node senses data it will always send it. Formally this property can be written as:

$$Source.SENSE\_DATA \rightsquigarrow Source.SEND\_DATA$$

The most important test is the *data transport property* that whenever a source model transmits the message (SEND_DATA), it leads to the sink model receiving that data (REC_DATA):

$$Source.SEND\_DATA \rightsquigarrow Sink.REC\_DATA$$

Note that the model allows transmission of only a single message so the above property can confirm the data transport check. The above properties are satisfied for all possible combinations of 5 nodes. By applying the above property to the flooding protocol, for all possible combinations of 5 nodes, topologies in which the source is not connected to the sink are eliminated. A successful verification of the above property in Uppaal is shown in Figure 4.5 and a trace generated by Uppaal in case the above property fails is shown in Figure 4.6(a). It is obvious that the property fails because although source node has sensed a data but the sink model remains in the LISTEN location. The sink model hasn't received transmitted message from any node that will move it to the REC_DATA location.

4.4.4 The Trace

The trace is converted into another graphical form as shown in Figure 4.6(b). The nodes are represented using circles annotated with the node ID. The sink or BS is represented by B, a source node is denoted by S, an attacker node is represented by A and normal nodes are represented by N. As the network contains multiple node models, these are represented using IDs Na, Nb,... or N1,N2,... etc. Messages are shown using arrows with the arrow tail indicating the message sender. The messages are numbered as 01,02, and message details are written separately. The message format employed in Uppaal is as follows: (Type, ID). Type is the message type (data in this case) and *ID* is the sender node's ID. This trace shows that there are four data messages exchanged between the nodes and data does not reach sink. Note that Uppaal automatically checks for all possible combinations when the property is tested for proof. In case the property fails in a message sequence, the trace is generated.

The research is based on checking the vulnerability of routing protocols against the DoS attacks, and DoS attacks are aimed at preventing data from the source reaching the destination (sink), there is no point in considering topologies where the source and the sink are not connected. For the analysis of future protocols such topologies are therefore avoided.

4.5 The TinyOS Beaconing Protocol

4.5.1 Protocol Description

We begin with an analysis of the TinyOS protocol [15], one of the simplest WSN routing protocols. This protocol constructs a spanning tree rooted at the BS. The data then flows from the source nodes back to the BS using paths in the spanning tree. Two types of messages are involved: hello *beacon* messages and *data* messages. A brief description of the protocol is as follows.

The BS periodically broadcasts a hello beacon which is flooded throughout the network.

$$B \rightarrow * : (beacon, ID_B)$$

A node N, on first hearing a hello beacon, makes the transmitting node its parent and ignores any future beacons. The node then rebroadcasts the beacon with its own ID:

$$N \rightarrow * : (beacon, ID_N)$$

This process is repeated throughout the network until the spanning tree is established. A source node S, upon sensing a significant event in its environment, unicasts the data back to its parent P. Data messages are unicast from the node to the parent node until the BS is reached.

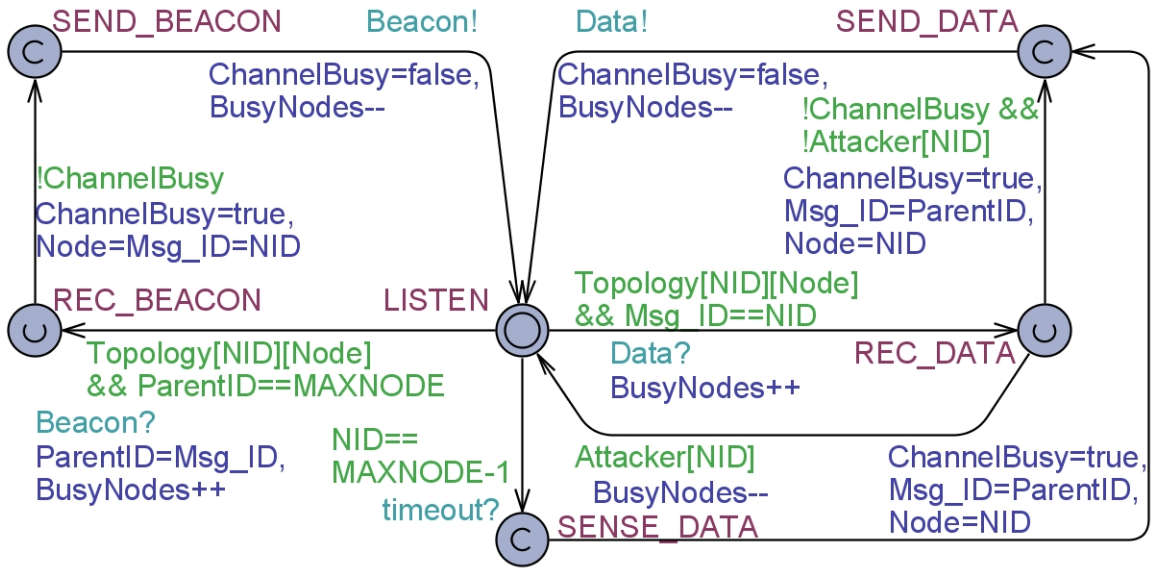
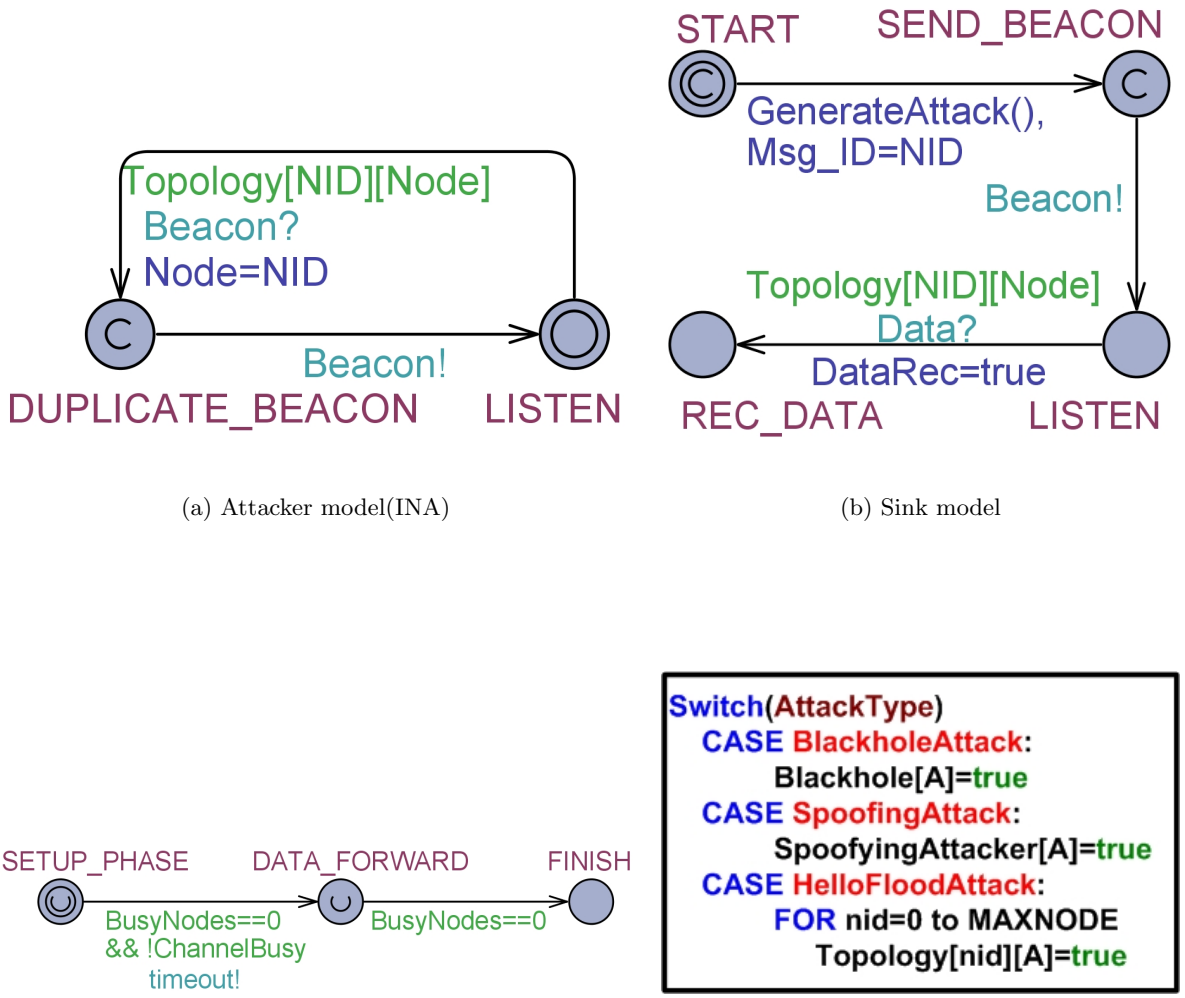


Figure 4.7: Node model for the TinyOS beaconing Protocol

$$S \rightarrow P : (data, ID_P, PAYLOAD)$$

4.5.2 Formal Model

The model used to evaluate the TinyOS beaconing protocol consists of an event generator model (Figure 4.8(c)), a sink model (Figure 4.8(b)), an attacker model and node model (Figure 4.7). The **event generator (EG)** model generates a `timeout` to move all the nodes from their setup phase (SETUP_PHASE) to their data forwarding phase (DATA_FORWARD). The EG model moves to the next phase only when variable `BusyNodes` becomes 0 indicating that all nodes become free (Section 4.3.4). The **sink** model consists of 4 locations. The START location is the initial location that is critical; so the sink model immediately moves to next location. The `GenerateAttack()` function is used to inform black hole attacker models that they are attackers (`Attacker` flag is true). The next location is SEND_BEACON after which the beacon message is generated in the next transition. The global variable, `Msg_ID`, is thus assigned the sink's ID. This location is again critical and thus is executed immediately on the next transition. The two critical locations are executed one after the other, preventing any other node model from modifying the `Msg_ID` variable. The sink model then moves to the LISTEN location and remains there until it receives a data message addressed to it. The `Topology` matrix is utilized to check the radio link connections between nodes for message reception. When data is received



(a) Attacker model(INA)

(b) Sink model

(c) Event Generator model

(d) GenerateAttack() function

Figure 4.8: Uppaal model for the TinyOS beaconing Protocol

by the sink, it moves to the FINISH location, sets the `DataRec` flag and the model deadlocks in this location.

The **node** model (Figure 4.7) starts in the LISTEN location and remains there unless it receives any message (data or beacon) or a `timeout` event from the EG model. The message is only received by a node model if the `Topology` matrix confirms that a link exists between the current node (`NID` variable) and sender node (`Node` variable). Upon receiving a beacon message (REC_BEACON), the node checks if this beacon is received the first time (`ParentID==MAXNODE`). If this guard is satisfied the sender node (Msg_ID) is assigned the parent (`ParentID`) of the current node model. The node then waits for channel to become free (`ChannelBusy` flag is false). In case the channel is free the node makes the channel busy (`ChannelBusy` becomes true), updates `Msg_ID` and the `Node` variable (Sender node ID) by assigning them its node ID (`NID`). As the global variables have been modified the next location is a critical one so that it is executed on the next transition. Finally, the node model trans-

mits a beacon (SEND_BEACON) and moves back to the LISTEN location decrementing the `BusyNodes` variable and clearing the `ChannelBusy` flag. The `BusyNodes` variable is incremented when the model receives the beacon and is decremented after the model becomes free. The same procedure is adopted when data is received (REC_DATA) by a node or if it senses new data (SENSE_DATA) enabled by a `timeout` generated by the EG model.

4.5.3 Verification

For the attacks considered, it was verified that data always reaches the BS from the source node, in all topologies where there is a path between them. This is achieved by writing a data transport property as was written for flooding:

$$Source.SENSE\_DATA \rightsquigarrow Sink.REC\_DATA$$

First the model was checked without any attacker model. It was confirmed that the above property is satisfied for all possible combinations of 5 nodes where the source and the sink are connected. This confirms that in the TinyOS protocol, in the absence of an attack, the data transmitted by a source will always reach the sink. Finally, the confirmation that attacks are successful in TinyOS is discussed in the next sections.

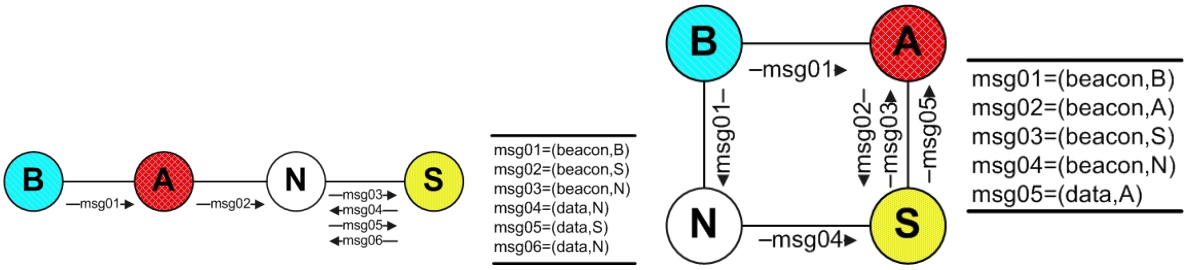
4.5.4 Black hole Attack

A black hole is modelled simply by having the node model forward beacon messages correctly but drop data messages. This is incorporated in the node model using a global flag, `Blackhole`, that enables the black hole attacker. The required property is violated and the trace confirms that when the attacker is a parent in the data forwarding path, the data may not reach the BS.

Figure 4.9(b) illustrates a successful black hole attack in one topology detected using Uppaal; there is a BS B, a source node S, an intermediate N and an attacker A. Note that here the node A behaves normally during beacon forwarding but drops the data once it becomes the parent of the source node S.

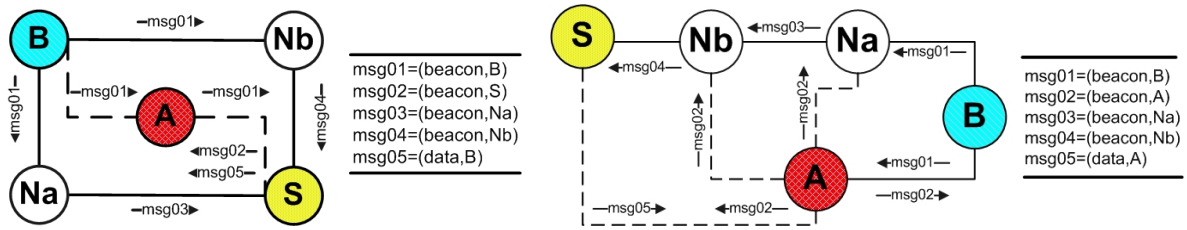
4.5.5 Sinkhole Attack

A sinkhole attack is modelled by having not only the BS, but also the attacker, initiate the broadcast of hello beacons. The attacker also ignores legitimate beacons from the BS. The attacker model is similar to the sink model shown in Figure 4.8(b). In this model, the data transport property is violated and the trace confirms that when the attacker acts as a sink hole, the source and intermediate nodes transfer the data to it rather than to the BS. Figure 4.9(b) illustrates one such trace detected using Uppaal in an analysis of one of all 4-node topologies. In this example, node A imitates a BS and broadcasts a hello beacon (msg01). The source node receives this beacon before the legitimate beacon (msg03) and so transfers the data to the attacker (msg04).



(a) Spoofing Attack

(b) Black hole and sinkhole attack



(c) Worm hole Attack

(d) Hello flood Attack

Figure 4.9: Traces for a topology in which the formal model detected successful attacks in the TinyOS protocol

For this simple protocol in such a small network, the scenarios illustrating the sink hole and the black hole attacks show identical results in their traces. However, the behaviour of the attackers is different in each case and in general, the scenarios illustrating the attacks will be different.

4.5.6 Wormhole/INA Attack

In an INA and wormhole attacks the attacking node is hidden and only forwards beacons from the BS to the source node. The attacker does not transfer data received from the tunnel, as the aim of a wormhole attack is to drop data transmitted in the tunnel. The INA model is shown in Figure 4.8(a).

Analysis of this model shows that the data transport property is not satisfied and the trace reveals that when a virtual connection (INA/wormhole) exists between the BS and a source

node, the source establishes the BS as its parent instead of one of the nodes within its range. Thus, when it transmits the data, it is lost as it is transmitted to a node that is reachable only via the virtual link. Figure 4.9(c) illustrates a successful INA in one of the topologies detected using the formal framework in an analysis of all 4-node topologies. Note that msg01 is replayed by the INA attacker A without adding itself, the attacker however does not forward the data (msg05). Similar results are obtained when the wormhole attackers were deployed instead of the INA.

4.5.7 Hello Flood Attack

The high power transmitter of the attacker in a hello flood attack is modelled by modifying the connection matrix to allow the attacker to establish unidirectional links with all the other nodes in the network; this is contrary to the usual assumption of symmetric links but accurately reflects the ability of the hello flood attacker.

Analysis of such a model shows that the required property is not satisfied and the trace reveals the cause of the problem. When the attacker broadcasts hello beacons it is heard by all the other nodes which then establish the attacker as their parent in the spanning tree. Data from any source node whose transmitter is not powerful enough to reach the attacker is lost. Figure 4.9(d) illustrates one 5-node topology, detected using the formal framework, that illustrates the problem. Notice that msg02 is received by all the nodes, including the source node S, which ignores the legitimate beacon from node N and establishes A as its parent. The transmission of msg05 does not reach the attacker due to the low power transmitter of the source node. Of course, it is trivial for the attacker in this case to act also as a sink hole if it so chooses.

4.5.8 Spoofing Attack

In a spoofing attack, the attacker transmits the hello beacon using the ID of the source node instead of its own. The data transport property is violated and the trace reveals that the problem is caused by the creation of a routing loop in topologies in which the source node and the attacker share a neighbour. Since the source node and its neighbour are established as each other's parent and data relays between them indefinitely. Figure 4.9(a) illustrates one such topology, detected using the formal framework from an analysis of all 4-node topologies.

4.5.9 Other DoS Attacks

The research has also studied the effects of the false injection, rushing and sybil attacks on the TinyOS protocol. In fact, the sink hole attack discussed earlier relies on the false injection of a hello beacon by the attacker. Hello flood enables the rushing attack. The sybil attack has been modelled by having the attacker transmit hello beacons using both its real ID and a

fake ID. The data transport property is not satisfied in this case and the trace showed that the neighbours of the attacker might establish a non-existent node as their parent and then attempt to forward the data to it. In the TinyOS protocol this attack resembles a spoofing attack.

4.6 The Authentic TinyOS Protocol using uTESLA

4.6.1 Protocol Description

The Authentic TinyOS protocol uses μTESLA [38] to authenticate messages. μTESLA is a lightweight authentication protocol aimed at reducing the computing resource requirements of nodes that implement it. This makes it practicable for use with the resource constrained nodes of a WSN. The Authentic TinyOS protocol is similar to the TinyOS Beaconing protocol considered in Section 4.5. The main difference is that the BS uses a μTESLA key, K_i, in the ith beaconing interval to generate a message authentication code (MAC) for the beacon. K_i is derived using a public hash chain whose seed is known only to the BS and thus no other node can generate or predict the next key. The BS starts by broadcasting the following message in first beaconing interval (t=0):

$$B \rightarrow * : (beacon, ID_B, 0, [MAC]_{K_{t+1}})$$

The receiving node N checks if the beacon has been received in the first beaconing interval. If this does not hold, the beacon is ignored otherwise it is pushed into a FIFO queue as the pair (ID,MAC) and the node broadcasts the beacon, using its own ID.

$$N \rightarrow * : (beacon, ID_N, 0, [MAC]_{K_{t+1}})$$

This process is repeated throughout the network. When the next interval begins, the BS discloses the key for the previous interval by broadcasting:

$$B \rightarrow * : (beacon, ID_B, K_{t+1}, [MAC]_{K_{t+2}})$$

This is again broadcast to all the neighbours and each node chooses the first authentic beacon sender as its parent and then ignores further beacons. Each node also retransmits the authentic beacon.

$$N \rightarrow * : (beacon, ID_N, K_{t+1}, MAC_{K_{t+2}})$$

4.6.2 Formal Model

It is not necessary to model the details of the authentication mechanism to undertake a useful formal analysis of the protocol. It is assumed that μTESLA provides a reliable authentication

mechanism. The focus of the analysis is on the effect of the attacks on the protocol, given this assumption.

In order to model the Authentic TinyOS protocol, the model of the TinyOS Beaconing protocol is modified so that the BS sends two beacons. The first with a null *Key* field and the second with a key that can be used easily to authenticate the previous beacon. The message format is now (Type, ID, Key, MAC). The MAC is modelled simply by using a number. The BS sends the first message as (id *beacon*, $B, 0, 1$) i.e. the MAC is 1 and the Key is null. Then the second message (id *beacon*, $B, 1, 2$) is sent, where the key 1 will authenticate the previous MAC and the new MAC is set to 2 for future use. Using this simple model, the whole protocol can be analyzed. Nodes check the authenticity of a beacon by testing if the MAC field of an entry (ID, MAC) in the FIFO matches the key received in the beacon. Beacon intervals are modelled simply by waiting until all nodes have received a beacon before a new beacon is transmitted by the BS.

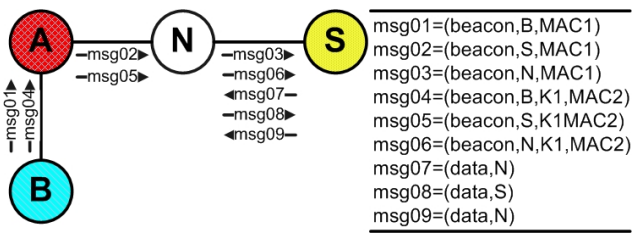
4.6.3 Performance of Authentic TinyOS against DoS attacks

Analysis of this model shows that the Authentic TinyOS protocol successfully resists the sink hole and the false injection attacks to which the simpler TinyOS Beaconing protocol was shown to be susceptible (Sections 4.5.5 and 4.5.9). The data transport property is satisfied, confirming that the data generated by the source node will always arrive at the sink in spite of the presence of these attacks.

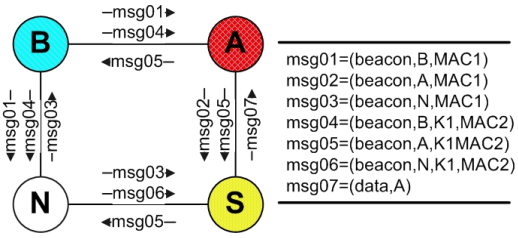
However, the analysis shows that several other attacks succeed, despite the use of authenticated broadcasts. Figure 4.10(a) shows a successful spoofing attack, Figure 4.10(b) shows a successful black hole attack, Figure 4.10(c) shows a successful wormhole attack and Figure 4.10(d) shows a successful hello flood attack. All of these scenarios were discovered using the counter-examples generated by the formal framework in an analysis of all 4-node and 5-node topologies.

4.7 Other Unsecured Routing Protocols

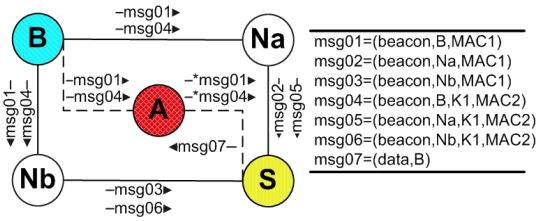
This research includes modelling and checking of other unsecured routing protocols such as Minimum Cost Forwarding (MCF) [41], Rumour Routing(RR) [39], LEACH [40], and Directed Diffusion(DD) [42]. As expected, they are susceptible to spoofing, black hole, sinkhole, hello flood, wormhole and sybil attacks. The rediscovery of these attacks strengthen confidence in our approach. These protocols have already been identified as susceptible to such attacks in several literature reviews. Moreover, some interesting results have also been obtained using the formal approach in the DD and RR protocols which had not previously been reported. These are discussed briefly below:



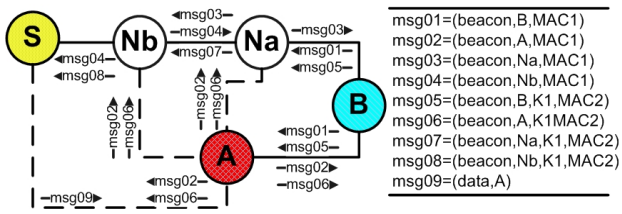
(a) Spoofing attack



(b) Black hole and sinkhole attacks



(c) Worm hole attack



(d) Hello flood attack

Figure 4.10: Trace for a topology for which formal model detected successful attacks in Authentic TinyOS (uTESLA) protocol

4.7.1 The Direct Diffusion Protocol

In Direct Diffusion(DD) [42], the BS flood (broadcast) interest messages with its node ID with the format:

$$B \rightarrow * : (interest, ID_B)$$

The node, upon receiving the interest, checks its cache (memory). If no match is found it creates an entry in the cache and rebroadcast the interest by replacing the ID with its own:

$$N \rightarrow * : (interest, ID_N)$$

The source S, upon receiving this message, sends a gradient message back to the node N from which it has received interest:

$$S \rightarrow N : (gradient, ID_S, ID_N)$$

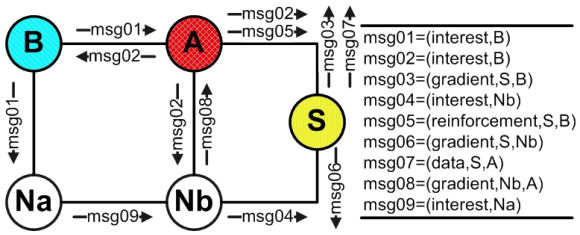


Figure 4.11: Traces for a topology in which the formal model detected successful sinkhole attack in the Directed Diffusion protocol

All the previous nodes forward this gradient until it reaches the BS, B. The BS then reinforces the first gradient it received by sending a positive reinforcement to N:

$$B \rightarrow N : (reinforcement, ID_N, ID_B)$$

It can also send negative reinforcement to cancel a route developed. The data is then forwarded back by the nodes into the reinforced path.

Consider Figure 4.11, showing a topology and trace of the Directed Diffusion protocol in the presence of a sink hole attack. Here the attacker behaves just like the BS. As soon as it hears an interest message from the legitimate BS, the attacker replays that message, identifying itself as the BS. Thus the source node will now send the data matching the interest message to both the attacker and the legitimate BS. The effect of a sinkhole attack was modelled by flooding an interest message from the attacker in which the ID is shown as the ID of the BS. Earlier research has suggested that the data in this case will reach both the attacker and the BS. But the formal framework trace in Figure 4.11 identifies a worse scenario in which sometimes the data may never reach the BS in the presence of a sink hole attack. From Figure 4.11, it can be observed that the attacker floods an interest message (msg02) as soon as it receives a legitimate one (msg01) from the BS. In this case, node *Nb* receives the interest message from the attacker (msg02) before the legitimate message from the node *Na* (msg09), which it then ignores. Thus, the gradient is set only towards the attacker, node A. In this way, a gradient is never established with the node *Na* and so the data will never reach the BS. Whenever the BS broadcasts its interest message again, the same process may be repeated.

4.7.2 The Rumour Routing Protocol

Rumour Routing [39] is a probabilistic protocol (Section 2.5.2) where queries are matched against events. Each node starts with a hello message informing neighbours of its ID:

$$N \rightarrow * : (hello, ID_N)$$

The neighbour tables are developed using that technique. After going through this process a source node of any event generates an agent. The agent contains the complete paths for all events, corresponding hop counts to each event, the nodes it has visited (to avoid loops) and time to live (TTL) in hops. The agent is sent to any neighbour (randomly) in the network; however, an unseen node is preferred by looking at the visited nodes field in the agent. The TTL is decremented at each node and the agent dies when the TTL becomes 0. Each node, upon receiving the agent, updates its 'Node Event Table' in memory as well as copying the complete path for each event. The format of an agent message is:

$$S \rightarrow N : (agent, ID_N, TTL, HopCount, Path[], Visits[])$$

Here S is the node sending the agent and N is a randomly selected neighbour node. Any node can generate a query for any event. The query has a similar format to that of the agent and also moves randomly until it reaches a node which knows the path to that event i.e. has been visited by the agent before.

The framework discovered an unreported bug in the Rumour Routing protocol [39] using the model-checking approach. The bug can be illustrated using the same topology as for the Directed Diffusion, shown in Figure 4.11. Consider the case in which a query has visited all the nodes before the agent has started. In that case, the agent arrives at each node after the query has been passed on. The exhaustive search made by the formal framework reveals the problem. Suppose in the topology in Figure 4.11 the query has taken a path of $B - Na - Nb - A - B - A$. As the node A has no unseen neighbours, it forwards the query again to the node B which forwards it back to A since it had sent it the last time to the node Na. Now suppose the agent has started and takes the path $S - Nb - Na - B$, the query and the agent will never meet and the data transport property is violated. It is extremely unlikely that this scenario will be detected using visual inspection even in smaller networks, but the formal analysis of all possible topologies of a 5-node network revealed it.

4.8 The Enhanced INSENS Protocol

4.8.1 Protocol Description

The Enhanced INSENS [19] is a modification of the Basic INSENS protocol to remove the danger of hello flood and rushing attacks as well as to reduce data traffic. INSENS is widely recognised as robust against certain DoS attacks. The basic INSENS [17] has 3 phases. In Phase 1 a route request is initiated by the BS; the neighbour table is routed back by all the nodes in Phase 2; and finally the BS sends the routing tables to all the nodes in Phase 3. Instead of 3 phases the Enhanced INSENS starts with a bidirectional verification before the BS initiates a route request. In this protocol each node, upon booting, establishes a pairwise key with its neighbours by broadcasting an echo containing a random nonce and its ID encrypted

using the global key, K_G, known to all the nodes:

$$N \rightarrow * : (echo, [ID_N, nonce_N]_{K_G})$$

The receiving node, R, replies with an echoback message containing the received nonce plus one and the pair key, $K_{R,N}$, between the nodes N and R; this is generated using a random number:

$$R \rightarrow N : (echoback, [ID_N, nonce_N + 1, K_{R,N}]_{K_G})$$

During the data forwarding phase, depending upon which node has the lower ID, either $K_{R,N}$ or $K_{N,R}$ is used as a key. The nodes delete the global key, K_G, when it has setup pair keys with all its neighbours. After this a node establishes a cluster key with all its neighbours. This is done by generating a random number, K_C, and unicasting it using the pair key:

$$N \rightarrow R : (cluster, [ID_N, K_{C_N}]_{K_{N,R}})$$

This cluster key is used later to broadcast a request beacon and thus INSENS uses this mechanism to prevent the hello flood and rushing attacks as the nodes will only accept requests generated by the known neighbours. The format of request message is:

$$B \rightarrow * : (request, ID_B, [OHC, ID_B, MAC_{K_B}^{request}]_{K_{C_B}})$$

The receiving node N first checks the sender ID. If it is found in its verified neighbour list it will process the message, otherwise it drops the message. Then the cluster key of the neighbour is used to decrypt the message and check if the hash is correct by iteratively applying the public hash function. This confirms that the message has originated from the BS. It then checks if the request is fresh by calculating the most recent element of the hash chain. If it is not fresh N drops the request. Otherwise it stores the hash and ID_B, appends the message with its ID and re-computes the message authentication code (MAC) by using its own key over the newly developed request and the MAC embedded in the received request message. Then it encrypts the message with its own cluster key, K_{C_N}, and broadcasts the request:

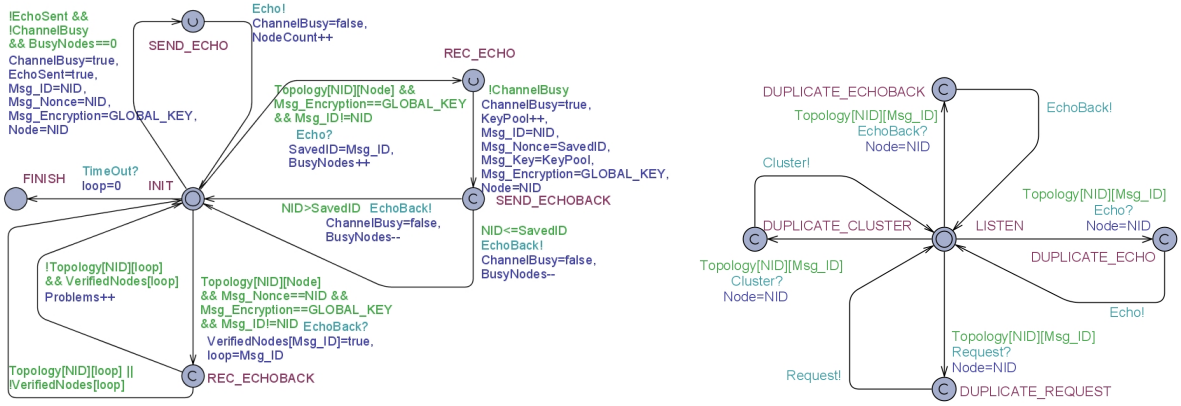
$$N \rightarrow * : (request, ID_N, [OHC, ID_B, ID_N, MAC_{K_N}^{request}]_{K_{C_N}})$$

where

$$MAC_{K_N}^{request} = (request, ID_B, ID_N, OHC, MAC_{K_B}^{request})$$

4.8.2 The Formal Model

The complete model is split into 4 parts: the node model, the event generator (EG) model, the sink model and an attacker model. The complete node model performs 4 phases:



(a) Node model for Phase 1 (Enhanced INSENS) (b) Attacker model (INA)

Figure 4.12: Uppaal models for Enhanced INSENS protocol

1. Pair keys are exchanged by echo beacons;
2. Cluster keys are unicast to all verified neighbours;
3. A request is flooded by the BS;
4. Data is unicast by the Source node.

For simplicity and to save state space, only the required phase of the node model is used when the attacker models are deployed. The node model for Phase 1 and Phases 3 & 4 are shown in Figures 4.12(a) and 4.13 respectively. The nodes exchange the messages of echo and echo back and using the same format as explained in the Section 4.8.1. All the locations are self explanatory in Figure 4.12(a) as in the names suggest these indicate sending/receiving either echo or echo back. Similarly, the states in Figure 4.13 depict sending/receiving the request or the data. The data is generated only by the source node. An attacker model (INA) is shown in Figure 4.12(b). The model shows that the attacker remains in the LISTEN location unless it receives any message (echo, echoback, cluster, or request) other than the data. It then retransmits that message without adding its own ID to the message fields.

The formal verification is used to test the following hypothesis:

"INSENS is vulnerable to the wormhole and the INA attacks in Phases 1&2 and also when the new nodes are deployed at a later stage. Moreover, a black hole attack is possible after the node is compromised or on success of the wormhole/INA at any stage."

The assumptions stated in Section 4.3.3 persist in this model. Additionally, it is assumed that the INA, the wormhole and the hello flood attackers are deployed before Phase 1; while the black hole attack can be deployed during any phase.

Section 4.8.3, Section 4.8.4 and Section 4.8.5.1 explain how the formal framework detects successful DoS attacks in INSENS and confirmation of these results using computer simulation.

4.8.3 Invisible Node Attack (INA)

4.8.3.1 Formal Analysis

All 5-node network topologies with 2 multiple paths were checked. The model confirmed that in most topologies data does not reach the BS in the presence of an INA even if a legitimate path exists. To check for additional multiple paths, a 9 node network was employed. As checking all possible topologies (2^{36} combinations) for 9 nodes is much more computationally demanding than for 5 nodes (2^{10} combinations), only a square grid node placement was checked. An error in a 9 node network was disclosed due to an INA even in the presence of 8 multi-paths. The property checked was the data transport property that after the source node has sensed the data it leads to the sink receiving that data:

$$Source.SENSE\_DATA \rightsquigarrow Sink.DataRec$$

The issue of the data not reaching the BS, due to the addition of unconnected nodes in the verified neighbour list of the source and the other nodes was subsequently investigated. In order to check the success of an INA one needs to model only the echoback part (Phase 1) of the protocol. The reason being INA and wormhole attack will remain unsuccessful if they are unable to create virtual links in the Phase 1. The messages are only accepted from verified neighbours after Phase 1. The node model (Phase 1) is shown in Figure 4.12(a). The check is a property claiming that a source node possesses fewer than N unconnected nodes in its verified neighbour list, i.e.

$$Source.FINISH \rightsquigarrow (Source.Problems <= N) \tag{4.1}$$

Here `Problems` is a local variable in the node model which counts the errors in the neighbour list in the node after Phase 1.

4.8.3.2 Simulation Results

The formal modelling has rigorously tested the INSENS protocol and has detected its susceptibility to INA. In this section the results are examined using computer simulation. Reasons for using computer simulation in the research includes checking scalability (large node networks) and to generate quantified results. The formal modelling checks each and every possible execution of a system but suffers from state space explosion in larger networks. Simulation not only allow replicating the results achieved by formal modelling but also allow checking bigger networks.

A fault in the Enhanced INSENS protocol was detected in the presence of INA in a particular topology using model-checking. To confirm this result the same network topology was tested using the TOSSIM simulator [229]. A complete protocol implementation was developed using the nesC [230] programming language. However, the encryption/decryption operations and the

message authentication code (MAC) were simplified. As the attacker need not know about the encryption details, the CBC mode technique was not used to generate the MAC (block cipher algorithm RCA [231]) as in the INSENS protocol. The emphasis of this research does not concern these encryption techniques so these were not implemented.

An ideal channel was used in the simulation in which no messages are lost as a result of RF noise. The reason for this decision was that the loss of messages as a result of an INA alone were to be examined. However, in the development of the Enhanced INSENS protocol the hidden terminal problem was faced. As most link layer protocols use CSMA/CA, the collision of messages is reduced but a problem arises when two nodes which are not within the range of each other, simultaneously transmit to the same node after detecting that the channel is free. In this situation the receiving node receives neither from the first node nor from the second. Physical node arrangements causing this problem are likely to be common in real networks. The 802.11 WiFi protocol provides an optional RTS/CTS protocol to reduce the hidden terminal collisions. However, 802.15.4, upon which many WSN are implemented, does not provide RTS/CTS. To avoid collisions due to the hidden terminal problem, a certain time slot is used for each node to send data depending on its node ID. The echo backs are also transmitted after inducing a random delay to avoid collisions. The other message transmissions are also adjusted so that the collisions were avoided. When this random delay was not employed in earlier experiments, it led to the loss of messages due to collisions and the results obtained were worse than the presented one. As the aim of this research was to find errors due to attack and cancel all other effects, these delays were introduced to avoid collisions.

The simulation were run both in the presence and the absence of INA. The attacker affected the data throughput in all cases, however, maximum damage was done when the attacker was not on the boundary of the grid. The data was sent periodically from the source to a single BS for a total of 1000 times and the BS maintained a record of the received data using the message ID attached to each message. Each experiment was repeated 20 times. The developers of INSENS do not explain how the data will flow towards the BSs. However, they do refer to forwarding data back to the nodes from which they received their first request beacon (parent nodes). In order to test the INSENS model for data forwarding three different methods were used to forward the data. In the first method, the source node randomly selected the next neighbour (if multi-paths are used) and then the data was subsequently sent back throughout to node parents. In the second method, the sender neighbour was always selected randomly. In the third method, each node adopts load balancing in forwarding data to its neighbours i.e. selecting all the neighbours before repeating any. Note that in the last 2 methods some data packets may take a long path and thus may collide with later data packets, e.g. for a 9 node network the average data delivery to the BS might reduce to 96% on average. But this data loss increased with the size of the network. For a 36 node network it was found that the data loss may reach 24% on average (76% throughput) in the absence of any attack and under ideal

conditions.

A 9 node network (3x3 grid), for which our formal framework detected a successful INA, was implemented. The simulation results confirmed that if the data was sent through the parent nodes, the data delivery percentage at the BS was reduced to 0%. This is because the links via an INA will always be marked as the parents (provides the shortest path) for all nodes within the source node's range. In the case of random selection, throughput was reduced to 10% on average [range: 4% to 17%] even if 8 multiple paths were used. (INA will enable 8 paths in this network). Thus, the simulation results confirmed that the formal framework correctly detected INSENS being vulnerable to the INA.

4.8.4 Wormhole Attack

It has been pointed out in the Section 4.8.3.1, that using the bidirectional verification phase of the model and using the property in equation 4.1, one can check in which topology a wormhole attack will have a detrimental effect. Later, networks of 25 and 36 nodes placed in regular grids of (5x5) and (6x6) were checked. It was confirmed that the Uppaal model always detected virtual connections. The same topologies were checked using TOSSIM as was done for the INA. It was confirmed that the data delivery rate reduced dramatically in the presence of a wormhole, even with multiple paths (up to 8).

A number of different tests were performed on a 36 node network (6x6 regular grid) with a density of 8 neighbours (non-boundary). There was one BS and one source in the network and both were placed at the opposite corners of the grid. The source node sent 1000 data packets to the BS and a total of 20 tests were performed. One end of the wormhole was placed near the BS while the other end was placed in the network near the opposite corner of the BS (near the source). It was confirmed that even if 4 multiple paths were available, only those nodes which were one hop away from the BS managed to transfer all their data correctly to the BS. In the case of parent selection (random selection for the first hop because of multiple paths), the percentage of data reaching the BS was reduced to 0%. Again, as with the INA, all nodes will mark the nodes via the wormhole tunnel as their parent.

If nodes are randomly selected, the average throughput of data reaching the BS was reduced to 78% on average [range: 75% to 80%] in the absence of an attack (1-4 multi-paths) and, surprisingly, to 2% on average [range: 0% to 6%] in the presence of a wormhole. Even in the absence of an attack, as nodes are selected randomly, the message transfer may take a very long time to complete and the collisions may occur at some point between the old and the fresh data packets. Thus, on an average, 20% of the data is lost because of randomness alone. Further throughput loss results from the selection of virtual neighbour in the data forwarding.

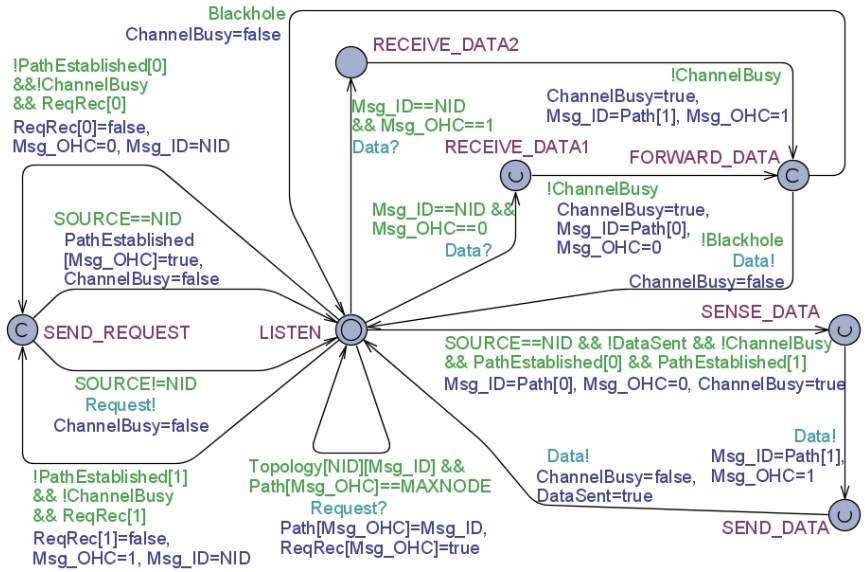


Figure 4.13: Node model for Phase III & IV (Enhanced INSENS)

4.8.5 Black hole Attack

Formal analysis and simulation have been applied also to an investigation of the black hole attack. The findings are reported in this section.

4.8.5.1 Formal Analysis

For the black hole attack, apart from replacing the attacker model, the node model (Figure 4.13) was modified (Phases 3 & 4 part). Similar to the wormhole study, the tests were performed using a subset of the complete protocol model containing only the related phase (request flooding and data forwarding phases in this case). This was checked for a 25 node network (5x5 grid) and with 2 BSs positioned at opposite corners. The data transport property was modified since receiving the data at either of the two BSs or sinks was acceptable:

$$Source.SENSE\_DATA \rightsquigarrow (\quad Sink1.DataRec \;||\; \quad Sink2.DataRec)$$

It was found that data transport property was violated when the number of attackers was increased to 2, each being placed near the individual BSs.

4.8.5.2 Simulation Results

The 25 node network checked above (Section 4.8.5.1) was implemented in TOSSIM in the presence of 2 and 4 BSs. The number of attackers and paths were also kept equal to the number of BSs for this network. A central BS was also added to the network to gather the data from all the BSs. For all these experiments 200 data packets were generated by the source node and 20 different tests were performed. Using random selection and then forwarding it to

the parent nodes, the average percentage of the data reaching the BS was 80% [range: 38% to 100%] (2 BSs; 2 attackers; 2 paths) and 55% [range: 44% to 72%] (4 BS; 4 paths; and 4 attackers). This percentage reduced in the presence of 4 BSs because 4 attacker nodes were deployed in this case.

The black hole attack was further tested using random and then the parent routing techniques with 4 BSs, 4 attackers and 100 nodes in 10x10 square grid. BSs were located one at each corner of the grid. Two series of tests were completed for this 100 node network:

- *Case 1:* First the source node position was kept constant in the middle of the network and the data was sent to all the 4 BSs a total of 1000 times. These tests showed that an average delivery percentage of 32% [range: 0% to 67%], 63% [range: 54% to 67%] and 100% for node density of 8, to 12 and then 20 neighbours for non-boundary nodes in the presence of 4 attackers (at each corner). However, the average throughput reduced to 5% [range: 0% to 16%], 42% [range: 0% to 63%] and 53% [range: 0% to 68%] in the presence of 8 attackers respectively.

- *Case 2:* In the second series of tests each node acted as a source but send data only once. With a density of 8 neighbours the average delivery percentage was 37% [range: 16% to 61%] and 5% [range: 1% to 18%] in the presence of 4 and 8 attackers respectively. However, by increasing density to 20 neighbours the average throughput improved to 80% even in the presence of 8 attackers.

As pointed out by the developers of INSENS in [19], the position of the BSs is influential, so this was placed at 4 opposite corners as was done in [19]. The source node is placed at the centre. We confirmed that the position of the attacker nodes strongly determined the reception of the data at the BSs. For example, in a 100 node network, four attackers present at a one hop distance from each BS caused considerable data loss. Similarly, when all the 4 attacker nodes were positioned near the source node, a significant data loss was observed. All attackers near one BS will have little effect as all remaining 3 BSs will receive all data and thus defeated the aim of the attacker.

The developers of INSENS claim that with 4 BSs and 4 multi-paths, data will eventually reach the BS as the density of WSN is normally high. It has been confirmed that although this is true, a high density cannot be guaranteed in all the networks and on every path. Therefore, even by reducing the number of neighbours to 8 and in presence of 4 attackers, the resulting loss is very high. Moreover, it has also been confirmed that, even with high density networks (20 neighbours), if the black hole nodes are chosen carefully (after observing data traffic) throughput may be reduced to about 15%. The reason being that some nodes transfer the data more than others and if these are chosen as attackers more data will be lost. Further simulation results on INSENS are provided in Section 6.7

4.9 Arrive Routing Protocol

4.9.1 Protocol Description

The Arrive protocol [44] has been developed to provide robustness against malicious and failed node attacks by observing the performance of neighbour nodes reputation. Unlike some other routing protocols, the reputation is maintained locally and is not shared with other nodes. Therefore, nodes cannot misrepresent their data. Also the packets are routed in multiple paths to prevent a single failure affecting the data to reach the BSs. The nodes not only forward the data designated to them to reputed nodes (Direct Participation) but also eavesdrop the traffic. In the case of a message failure in the neighbourhood (a neighbour does not forward message after receiving a message from another neighbour), the node forwards the same message that is not addressed to it (Passive Participation). Although this method creates extra traffic, it provides robustness.

The Arrive protocol uses first a breadth technique to assign nodes a level as in the TinyOS beaconing protocol [15]. The BS has a level of 0, the nodes one hop away from the BS have a level of one and so on. The authors have not explained this issue in detail but it is expected that a message of the following format is initiated and broadcast by the BS 'B':

$$B \rightarrow * : (level, ID_B, L_B)$$

Here 'level' is the message type, followed by the sender ID and the level L attached. The BS initiates with the level 0. A node upon receiving a lower level updates its current level by adding one into the received level. It then rebroadcast the level message with the new level L:

$$N \rightarrow * : (level, ID_N, L_N)$$

Moreover, if a node receives a lower level beacon it will mark the sender as its *parent*. On the other hand, senders with the same level are assigned as *neighbours*. The sender nodes with higher levels are ignored. Note that the neighbour definition in Arrive is different from usual definition used in this thesis. Once the setup phase is completed, a source node S unicasts its data to either its parent or a neighbour depending upon the node's reputation. A neighbour node with a reputation lower than a certain threshold is not selected. The data message format is:

$$S \rightarrow N : (data, ID_N, [ID_S, Event_E])$$

The data message contains the source ID and event number E. A node N upon receiving this message again forwards it to either the parent or the neighbour:

$$N \rightarrow P : (data, ID_P, [ID_S, Event_E])$$

In case a node hears data sent to one of its parents or neighbours by another node, it acts as a *passive forwarder* and broadcasts the data if either of the following two conditions exist: (i)

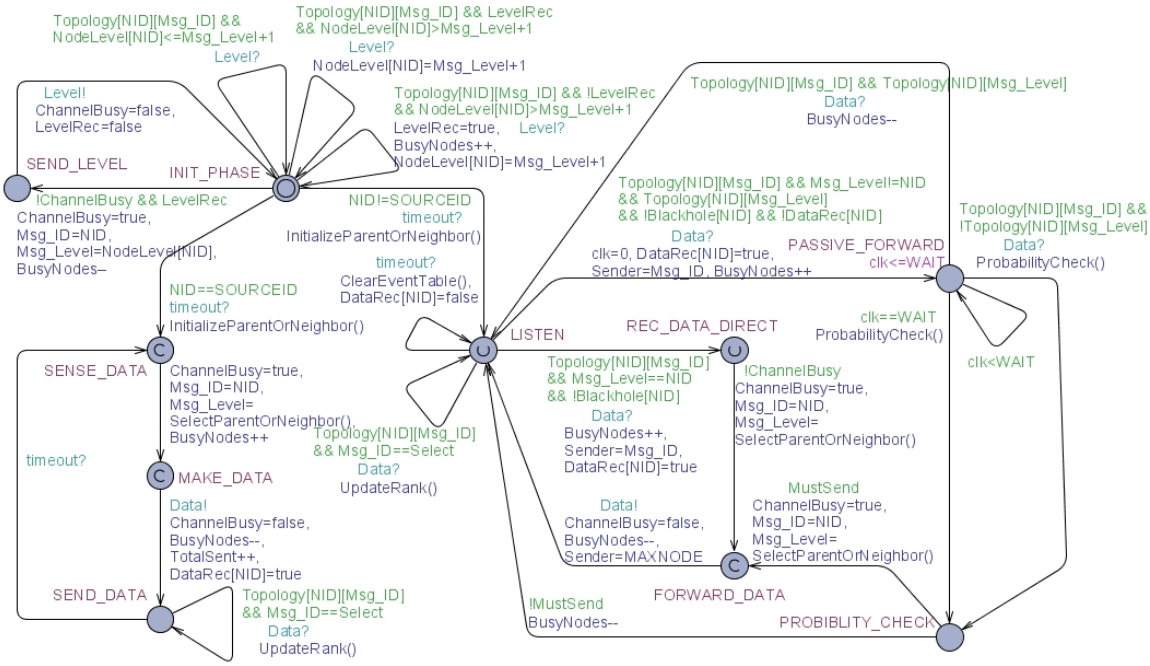


Figure 4.14: Node model for the Arrive Protocol

the receiver node has either not forwarded the data or (ii) forwarded data to a node not in that node's radio range. The developers of Arrive, however, warned that the Passive participation must be used with caution because unidirectional links and hidden terminal problems may cause the passively participating nodes to act erroneously.

4.9.2 Formal Model

4.9.2.1 Model

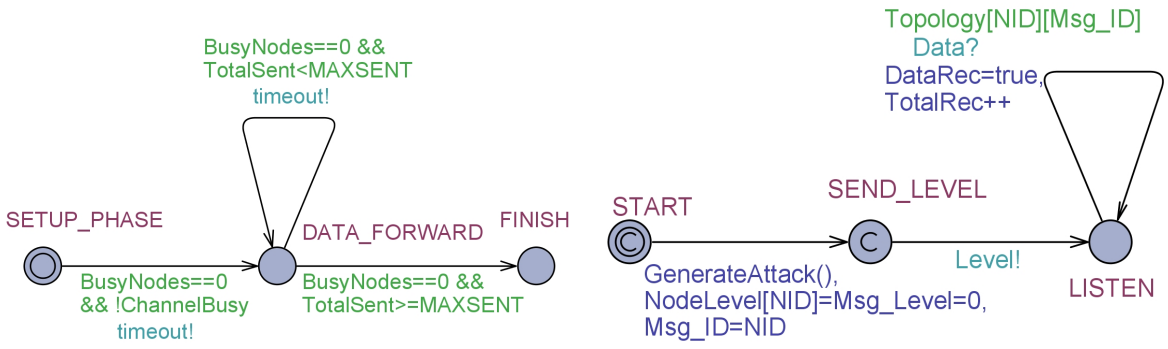
The message format employed for Arrive in Uppaal is as follows: (Type, Sender ID, Level Sent) for level message and (Type, Destination ID, Source ID, Event ID) for data message. *Type* is the message type (i.e. level or data); *Level Sent* is level broadcasted by a node; *Source ID* is the Source ID and *Event ID* is the unique Event ID for each event; and the remaining fields are self explanatory.

The complete Uppaal model is split into 4 parts: the node model, the event generator (EG) model, the sink model and an attacker model. In Arrive, the wormhole and INA ***attacker models*** are developed by creating a unidirectional link between the sink and source node. This was done to generate a tunnel between these, which only forwards level messages from the sink to the source. Note that the tunnel does not transfer data because the source can hear the sink but opposite is false. This modification in attacker model is done to save state space as larger networks are checked in this protocol.

The ***node model*** is shown in Figure 4.14. All nodes start from the INIT_PHASE, the setup phase. A node waits for a level message and upon its reception it updates its level. It is worth noting that all the node levels are initialized to a very high value and get updated only if a lower level value is received. There is a flag `LevelRec` in the model which is set when a level beacon is received and is cleared when the level beacon is broadcasted (SEND_LEVEL). A variable `BusyNodes` indicates how many nodes are busy receiving or sending messages. The EG model reads this variable and moves to the next phase when this variable becomes 0. All the node models leave the setup phase and move to the data forwarding phase when the EG model triggers them using the `timeout` message. This broadcast message is used to indicate an event to all the nodes simultaneously. All nodes initialize their parents and neighbours using the `InitializeParentOrNeighbor()` function which utilizes the topology matrix and the levels received in the setup phase. If a node is a source node, it senses data (location SENSE_DATA) and broadcast (location SEND_DATA) it. Another function `SelectParentOrNeighbor()` is used to select a neighbour before the data is forwarded. Data for multiple paths can be generated by sending more than one data message in this part of the model.

If a node is not the source, it will move to the LISTEN mode and will expect data messages. Upon receiving data addressed to it (REC_DATA_DIRECT), a legitimate node forwards it (FORWARD_DATA). If the data is not addressed to a legitimate node (`Msg_Level != NID`), it checks if the data has been addressed to its neighbour/parent (`Topology[NID][Msg_Level]` is true). If so, the node moves to the passive forwarding phase (PASSIVE_FORWARD) and remains in this phase for a certain time (10 clock cycles). The model checks two conditions here: (i) Data has been forwarded by the observed node during this time and (ii) Data has been forwarded to a neighbour within the range of current node, i.e. `Topology[NID][Msg_Level]` is true. If any of the above conditions is not satisfied and a certain time has elapsed (DELAY), the node model checks the probability to forward the data (PROBABILITY_CHECK) using a global function `ProbabilityCheck()`. If the probability is higher than the threshold value, the node model will forward data (FORWARD_DATA) otherwise it moves back to the LISTEN location. Another local variable `Sender` is used to track the sender ID of the data message in order to ensure that the data is not sent back to it.

The ***event generator model*** is shown in Figure 4.15(a). The EG model generates the event `timeout` to indicate the source nodes to sense data from the environment. The model starts with the SETUP_PHASE. When all the node models complete the setup phase (`BusyNodes` becomes 0), the nodes are triggered to move to the operation phase by sending their corresponding timeout to all the node models. This message also triggers the source to initiate the data message. The model tracks the data flow and when all the nodes become idle (`BusyNodes` is 0), it generates a new timeout message to enable the source node to generate a new data message. A variable `TotalSent` is incremented each time the source generates a new message and thus tracks the total number of messages generated by the source. When all messages have been



(a) Event Generator model

(b) Sink model

Figure 4.15: UPPAAL models for the Arrive Protocol

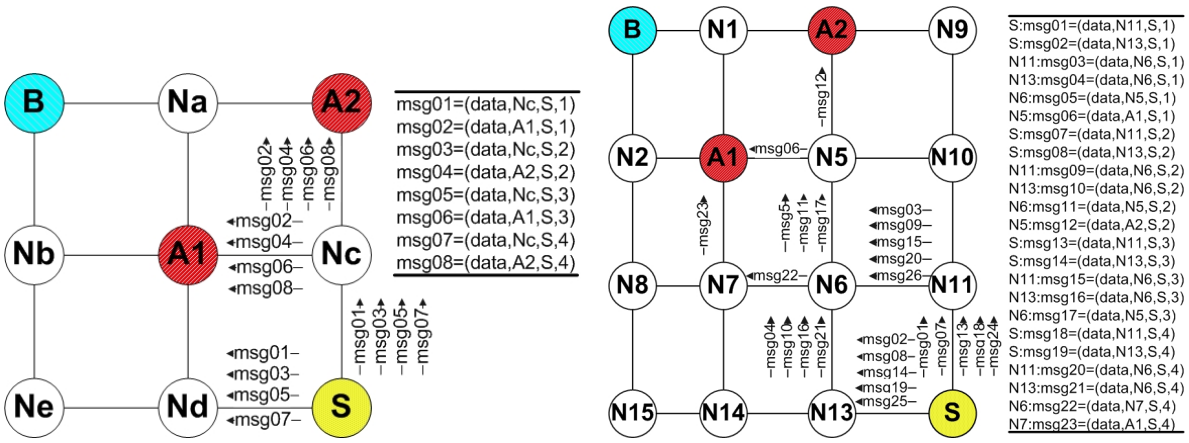
sent, the EG model moves to the FINISH location. The maximum number of data messages generated is limited by using the `MAXSENT` constant to reduce the state space.

The ***sink model*** is shown in Figure 4.15(b). The sink model immediately moves out of the initial location (START). The function `GenerateAttack()` is used to generate different attacks if needed. The sink model then broadcasts the level message (SEND_LEVEL) with level 0. Once this is done, the model remains in the LISTEN location with a self loop. This loop is only triggered if a node within the radio range broadcasts the data. The flag `DataRec` is set even if a single data message is received. A global variable `DataRec` keeps track of the total number of data messages received by the sink.

4.9.2.2 Verification

The properties checked include following claims:

1. All nodes will broadcast a level message
2. The source node eventually senses and broadcasts data
3. The Arrive protocol will always finish its setup phase
4. The Arrive protocol will not deadlock
5. All nodes attain the correct level
6. Neighbours and parents are assigned correctly
7. The BS will receive at least one data message from the source
8. The BS receives a number of data messages from the source



(a) Black hole attack with a single path

(b) Black hole attack with multiple paths

Figure 4.16: Trace of Uppaal showing that black hole is possible in Arrive

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.1.1. Next, different attack models have been applied to the Arrive protocol to check whether the properties were still valid in the presence of these attacks. In the case of any property failing, the Arrive protocol is considered susceptible to that particular attack and Uppaal automatically generates a trace. This trace shows how an attacker has succeeded. In the next sections we will discuss these attacks and the extent of their successes or failures.

4.9.3 Black hole Attack

4.9.3.1 Black hole Attack with a Single Path

For networks up to 5 nodes, it was confirmed that the Arrive protocol successfully defeats a single black hole attack. However, a problem arises when there are 2 black hole attackers within any node's range especially when there is no legitimate parent/neighbour of that node. Figure 4.16(a) shows a trace generated by Uppaal in which the properties fails in the presence of 2 black hole nodes. In this trace the level propagation phase is excluded because the attack is launched in the data propagation phase.

Note that the 2 black holes do not need to collude (within one another's range). The source nodes keeps on sending data to node Nc as it always eavesdrops to check if the node Nc has forwarding the data to A1 or A2. This also improves the ranking of Nc. Node Nd will not forward data passively as it has not eavesdropped the data sent to any of its neighbours (Nc and Nd are not the neighbours). Node Nc first attempts to forward data to node A1. A1 does not forward the data, so the ranking is lowered and at the next time Nc selects A2 for data forwarding. Unfortunately A2 is also an attacker node so the data is lost again. Nc lowers the ranking of A2 and tries to transmit the data to A1 and this process continues. Note that even the presence of another node having a lower level than node Nc will not help here because the nodes only forward data to their neighbours (same level nodes) or parents (lower level nodes).

We later confirmed this by employing 4x4 and 5x5 grids instead of 3x3 grid. Thus a flaw in the Arrive protocol was detected through the use of our framework. Even if the nodes A1 and A2 are dead, the same error is still possible. Note that a route does exist between S and B via S-Nd-Ne-Nb-B. Any other node's message forwarded by the node S (higher level nodes in a larger network) will be lost as well because this will also follow the same path.

4.9.3.2 Black hole Attack with Multiple Paths

In Section 4.9.3.1 even if Arrive uses 2 or more disjointed paths to transmit the same data, the data may eventually reach the BS. However, in the case of 2 paths, the source node S sends half the packets through node Nd, the other half that goes through Nc will be lost and Arrive cannot detect it. To confirm this finding and to model the disjoint multipath technique used in Arrive, a larger network of 16 nodes was considered. The node model presented earlier was modified so that the source nodes can unicast data using multiple paths. In the function `SelectParentOrNeighbor()` another flag array `Sent[]` is introduced which becomes set when data is sent to a neighbour. The flag is then cleared when the event generator model generates a `timeout`. A function `ClearEventTable()` clears the `Sent` array. Thus all flags are cleared before a new event is generated by the source. A simple flag is used to model the whole event table to reduce state space. This ensures that the same neighbour node is not selected in data forwarding for a single event as intended by Arrive. This had not been modelled earlier (Section 4.9.3.1) because then only one path had been considered.

The properties are violated again as shown by the trace of Figure 4.16(b). The message format (Type, Destination ID, Source ID, Event ID) in the trace is amended this time and the Sender Node's ID is added at the beginning of the message. This has been done to make the trace more readable. Moreover, to simplify the process, the level beacons are omitted from the trace and only those messages are placed in the trace which are directly received.

In the Arrive protocol a node only stops forwarding to a neighbour/parent if the reputation or ranking is below a threshold. This value was set to 1 in the model. Therefore, a node will not select a neighbour/parent even if it drops a single packet. Still the data transport property fails for N up to 4. The trace confirms the path taken by the nodes. Although the data is sent to 2 disjoint paths via nodes N11 and N13, both the data packets are eventually lost.

An interesting point to note here is that the nodes use reputation in the model after a single message loss. However, this is not actually the case in a real scenario and many data packets may be lost before the black hole nodes are isolated. Moreover, this is a small network and for larger networks, more routes can pass through the black holes resulting in further loss of the data. As explained in Section 4.9.3.1 passive forwarding is also not efficient here.

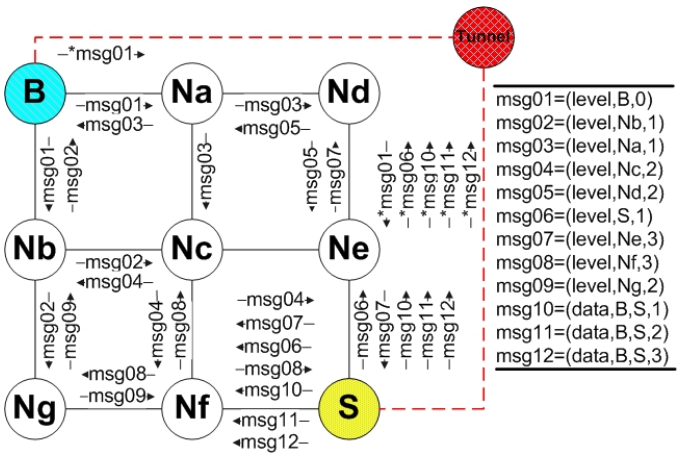


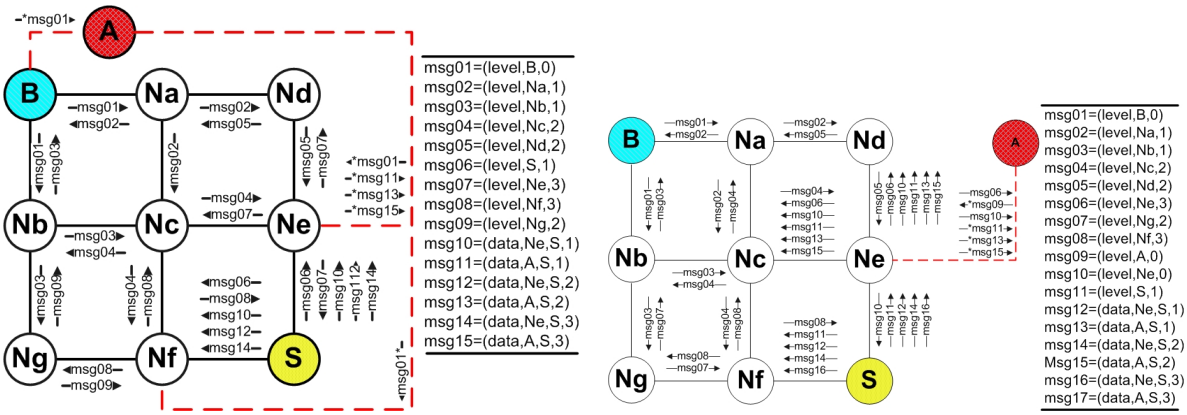
Figure 4.17: Trace of Uppaal showing that wormhole is possible in Arrive

4.9.4 INA

The formal-checker confirmed that INA is unsuccessful in Arrive. Although the nodes gain an incorrect level, the data transport property (Claim 8) holds confirming that the data is received by the sink. The only side-effect of INA is that the nodes next to the INA will get a level lower than the true one, their parents become neighbours, and their neighbours are removed. Moreover, INA creates virtual parents. However, INA still remains unsuccessful because when a node forwards data to its virtual parent (2 levels away) the nodes which are one level away from the sender node perform passive forwarding so the data is eventually forwarded. The ranking of the virtual parent is lowered resulting in excluding the INA for future data forwarding. The developers of Arrive seem to have been unaware of this fact and do not claim this behaviour. The formal framework, however, has proved that the passive forwarding fails in INA.

4.9.5 Wormhole Attack

Figure 4.17 shows a trace generated by Uppaal in which the data transport property fail in the presence of a wormhole attack. Here the tunnel indicates two connected wormhole nodes exists between the nodes B and S. One attacker is within the radio range of B and other is next to S. Thus nodes B and S are virtually connected. Thus the node S gets a level of 1 and the nodes which should be its parents (Ne and Nf) get a level 2 which is higher than that of node S. The messages msg01 to msg09 show the setup phase and how the nodes acquire their levels. When the source node S sends data, it has only one parent node B which is not within its radio range. Therefore, whenever a node transmits the data to B, the node ranking of B is lowered. But as there is no other node (neighbour/parent) that can be chosen to be the next hop, the data is always sent to B (via virtual link) and thus gets lost. Moreover, the nodes Ne and Nf cannot act as passive forwarders. The reason is that the necessary condition for a passive forwarding (i.e. B is within their range) is nonexistent. This clearly shows that the wormhole attack can



(a) Hello flood attack

(b) Sinkhole attack

Figure 4.18: Trace of Uppaal showing other DoS attacks are possible in Arrive

succeed in the Arrive protocol.

4.9.6 Other DoS Attacks

It has also been confirmed that the Arrive protocol is susceptible to many other DoS attacks such as hello flood, rushing, sink hole, spoofing attacks etc. It has been confirmed that if the *hello flood* attacker's transmissions reach all the network nodes (unidirectional), then the hello flood eventually fails in the Arrive protocol. Since all nodes have the same level, passive forwarding will enable the data to eventually reach the sink. However, as the probability of passive forwarding should be low [44], the hello flood attack can cause considerable message loss before being detected. Our formal analysis confirms that, if all the nodes are not within the attacker's radio range, the hello flood is successful in the Arrive protocol (Figure 4.18(a)). In many cases, only a few nodes will receive these unidirectional transmissions. Our formal framework has confirmed that in one such case, the nodes get an incorrect level and data will not reach the sink (data transport property fails).

A trace shown in Figure 4.18(b) confirms that both *spoofing* and *sink hole* attacks are successful in the Arrive protocol especially when the attacker node behaves as if it is a BS by sending a level of 0 (msg09). Even if the attacker is within the radio range of the other nodes, passive forwarding will not be adopted by any node, assuming that this node is the BS. It has also been confirmed that the spoofing attack is successful since any node can spoof level of 0 and thus can disrupt the whole network.

In a *rushing attack*, an attacker transmits incorrect messages, prior to the legitimate ones, with the aim that the nodes reject the legitimate messages and accept the incorrect messages. The successful wormhole, sinkhole and hello flood attacks in the Arrive protocol make the rushing attack possible. The nodes accept incorrect messages in the level propagation and reject the correct level beacons.

4.10 ARAN Routing Protocol

4.10.1 Protocol Description

ARAN [45] uses public key cryptography to ensure the integrity of routing messages. It is based on finding the quickest paths instead of the shortest ones. This means that ARAN avoids hop counts to discover routes. Initially, a source node S begins a route discovery process by broadcasting a route discovery message identifying the target node T:

$$S \rightarrow * : (RDP, T, cert_S, nonce_S, t, sig_S)$$

Here RDP means that this is a route discover phase, S and T are the identifiers of the source and the target, respectively, $nonce_S$ is a nonce generated by node S, t is the current time-stamp, $cert_S$ is the public-key certificate of the source, and sig_S is the signature of the source on all of these elements. Later, as the request is propagated in the network, intermediate nodes also attach their signatures. Therefore, upon receiving an RDP, a node A transmits the following message:

$$A \rightarrow * : (RDP, T, cert_S, nonce_S, t, sig_S, cert_A, sig_A)$$

When a neighbour of A (e.g. B) receives this route request, it verifies the signatures and the freshness of the nonce. If the verification is successful, then B updates its routing table for the source node S with A being the next hop node. Node B then replaces the certificate and the signature of A with its own and rebroadcast:

$$B \rightarrow * : (RDP, T, cert_S, nonce_S, t, sig_S, cert_B, sig_B)$$

A target node T, upon receiving the first route request verifications, updates its routing table as done by the previous nodes. T then unicasts a route reply message (REP) to source node S in the reverse path (node B) of the discovered route:

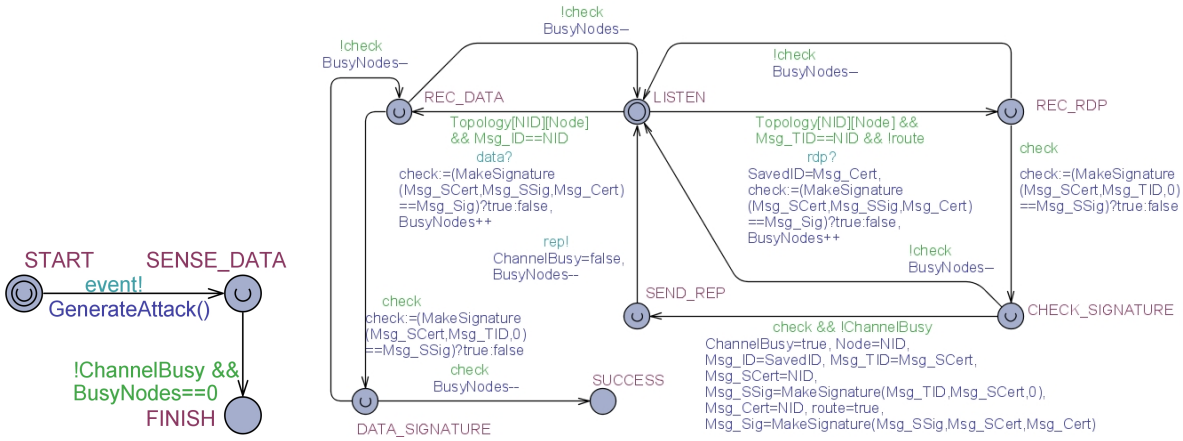
$$T \rightarrow B : (REP, S, cert_T, nonce_S, t, sig_T)$$

Here $nonce_S$ and t are the nonce and the time-stamp obtained from the RDP message, respectively. S, $cert_T$ and sig_T are the identifier of the source, the public-key certificate of T and the signature of T on all of these elements, respectively. Similar to RDP, REP is also signed by the intermediate nodes, as well. Hence, the route reply sent by B to A is:

$$B \rightarrow A : (REP, S, cert_T, nonce_S, t, sig_T, cert_B, sig_B)$$

Upon receiving the REP, the node A verifies both signatures. If both are valid, A forwards the REP in the reverse path after replacing the certificate and the signature of B, with its own, in the message:

$$A \rightarrow S : (REP, S, cert_T, nonce_S, t, sig_T, cert_A, sig_A)$$



(a) Event Generator model

(b) Target model

Figure 4.19: Uppaal models for the ARAN Protocol

Node A also updates its routing table for the target node T with B being the next hop. Although, nothing is stated in published protocol regarding data propagation we believe it has the same pattern as that of REP. For example, suppose the source node is sending data to target T via the verified path starting with node A:

$$S \rightarrow A : (DATA, T, cert_S, nonce_S, t, sig_S)$$

The nodes then again communicate the data in the same pattern:

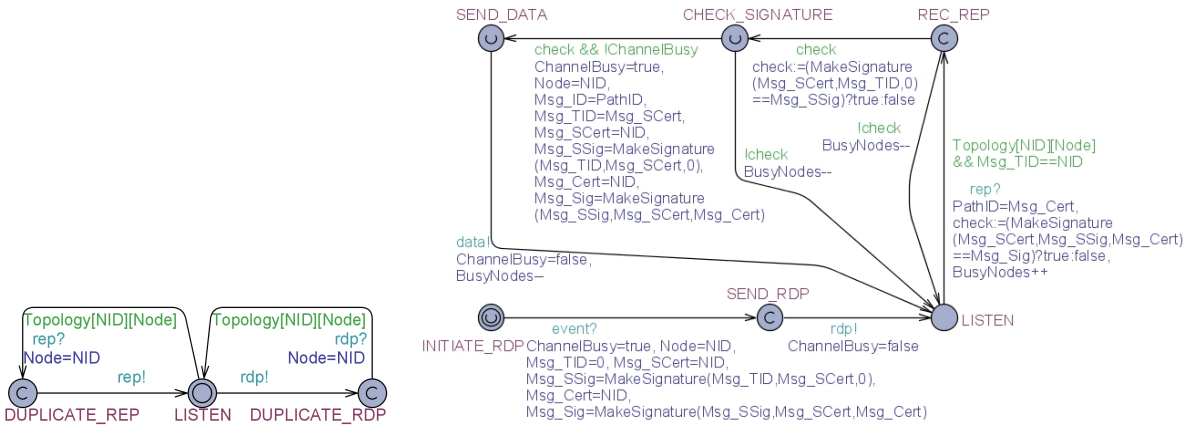
$$A \rightarrow B : (DATA, T, cert_S, nonce_S, t, sig_S, cert_A, sig_A)$$

4.10.2 Formal Model

It is believed that DoS attacks such as black hole, hello flood and wormhole attacks are still possible in ARAN protocol and can lead to a significant loss of data. Therefore, the formal framework is designed to detect ARAN's vulnerability to DoS attacks. While modelling ARAN it is assumed that the channel is ideal i.e. no message is lost because of collision or noise; the nodes are placed in a rectangular grid and have the same radio range; and the density of network is limited to 4 maximum neighbouring nodes.

4.10.2.1 Model

The message format employed in Uppaal is as follows: Type, ID, TID, Source/Target Certificate, Source/Target Signature, Node Certificate, Node Signature. Type is the message type (i.e.



(a) Attacker models (INA)

(b) Source model

Figure 4.20: Uppaal models for the ARAN Protocol

RDP, REP or data); ID is used to address the nodes in unicast messages (RDP or data) and remains null in broadcast (RDP); TID is the target ID to which the message is addressed (target in RDP/data and source in REP). The remaining fields are self explanatory. The nonce field and the time stamp are not modelled since it has been assumed that these will remain unchanged throughout due to public key signature security, saving state space. Therefore, the source signatures are based on TID and Source Certificate which serves the purpose of the encryption in the model. The complete Uppaal model is split into 5 parts. These are the event generator model; attacker model; target model; source model and node model.

The ***event generator model*** has the task of generating different events in the protocol as shown in Figure 4.19(a). The event generator starts in the START location and generates a sense data event from the environment (SENSE_DATA). The nodes are informed of these events through the message and it models as a trigger that enables a node to do a specific job. Upon receiving this message the source node generates an RDP. The function `GenerateAttack()` creates attacks like black hole attack etc in the ARAN model. If all the nodes become free along with the channel the event generator moves to the FINISH location.

The ***attacker model*** comprises different attackers. For the black hole attack the model is a simple node that does not forward data, the attack model differs for other attacks. The INA model is shown in figure 4.20(a).This attacker simply duplicates the message it receives either RDP (DUPLICATE_RDP) or REP (DUPLICATE_REP) without adding itself to the route. The attacker does not forward any data. A wormhole attacker is modelled by using a separate message tunnel that models the tunnel which lies between 2 wormhole nodes.

The ***source model*** is shown in Figure 4.20(b). The source node starts in the INITI-

ATE_RDP location and builds the RDP message when the event generator triggers an `event` message. It then broadcast RDP (SEND_RDP) to its neighbours. Note that the *certificate* field is replaced by the *node ID* in the models. The signature is then calculated using a global function `MakeSignature()` which simply adds the variables passed to it. The source node then moves to the LISTEN phase and waits for the reply, REP. Upon receiving the REP message (REC_REP) it checks if the signature is similar for both the sender and the target nodes (CHECK_SIGNATURE). If any of the 2 signatures are incorrect the model moves back to the LISTEN location. If both the signatures are correct, the source node unicasts (SEND_DATA) the data message. A variable `PathID` has been used to save the ID of the first node which had authentically sent the reply message. The data is then unicast to that node and the source node again moves back to the LISTEN phase.

The ***target model*** is shown in Figure 4.19(b). The target node (or sink) starts in the LISTEN location. Upon receiving an RDP message, it checks three conditions: (i) that the message is addressed to it (ii) the signature of the sender is correct and (iii) that the signature of the source node is correct (CHECK_SIGNATURE). If all three conditions are met, the target node replies in the reverse path (stored ID in `SavedID` variable) by sending the REP message (SEND_REP). The target adds its own signature similar to what the source node had done initially while sending an RDP message. It then moves back to the LISTEN location. Upon receiving the data message, the target again performs two tests i.e. checking the sender node's signature and the source node's signature (DATA_SIGNATURE). If both the signatures are correct, the target model moves to SUCCESS location, otherwise it moves back to the LISTEN location.

The ***node model*** is shown in Figure 4.21 and models all the intermediate nodes that are neither the sources nor the targets. The node model starts in the LISTEN location and upon receiving any message (RDP, REP or data) it checks if the sender node's signature is correct. If the signature is correct, then the node rebroadcasts that message by adding its own signature. It is worth noting that the signature and the certificate of source/target pair as well as the Target ID remains unchanged in all the cases. A variable `SavedID` stores the first node from which the RDP was received so that it can later send REP to it. Another variable `PathID` stores the first node which sends back the REP, so that data can be sent to it later on.

4.10.2.2 Verification

The claims/properties checked for ARAN protocol were:

1. Nodes always rebroadcast the RDP message
2. Nodes always rebroadcast the REP message
3. Legitimate node forwards the data fairly

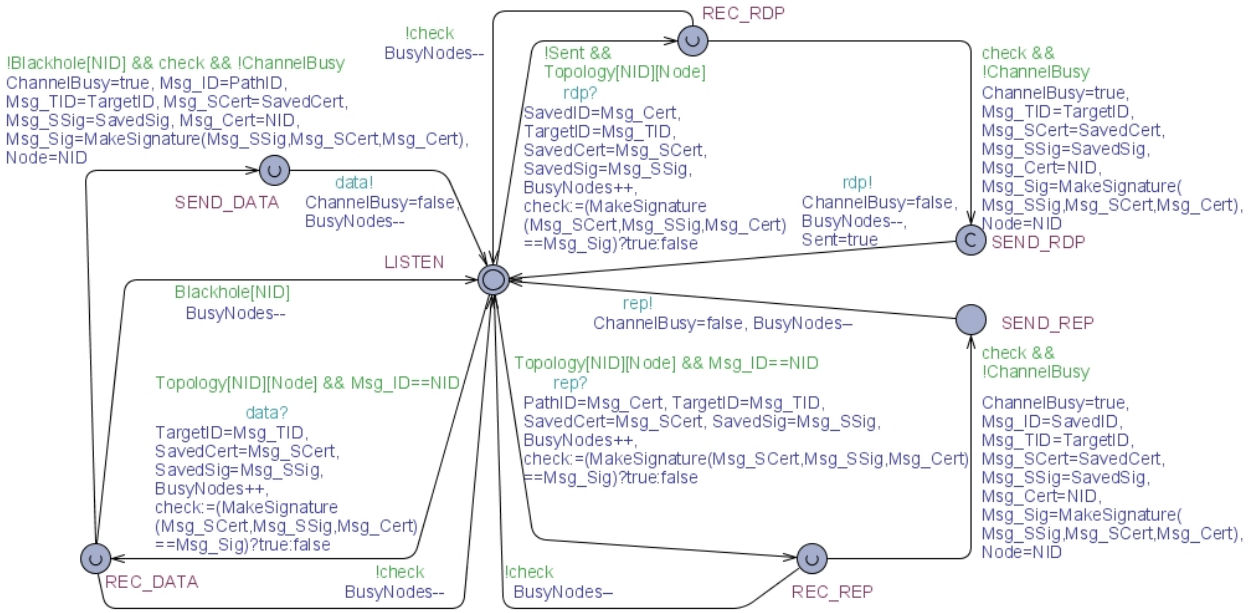


Figure 4.21: Node model of the ARAN Protocol

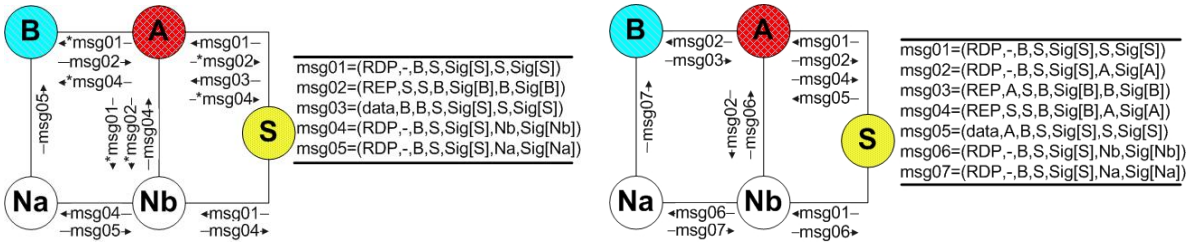
4. No deadlock in the ARAN protocol

5. Data from the source node always reaches the target node

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.1.2. By applying different attack models to the ARAN model we have been able to confirm whether or not the property that checks the successful data transport property (Claim 5) is still valid in their presence. In the case of a data transport property failure, ARAN is considered susceptible to that particular attack.

4.10.3 Worm Hole Attack/INA

Figure 4.22(a) shows a trace generated by Uppaal in which the data transport property (Claim 5) fails in the presence of INA. Here the node A is INA and retransmits the messages msg01, msg02 and msg04 which it eavesdrops from its neighbours without adding itself. Note that the node A does not duplicate data message (msg03). Thus, it creates a virtual link between the nodes B and S. Node S then transmits the data to node B which was not forwarded, thus causing the data transport property to fail. A similar behaviour was observed when the *wormhole* attacker was employed. However the legitimate RDP message (msg05) was rejected by the target node B as it had arrived late. This must also have enabled the *rushing attack*.



(a) INA/Wormhole Attack

(b) Black hole attack

Figure 4.22: Trace of UPPAAL showing other DoS attacks are possible in ARAN

4.10.4 Black Hole Attack

Figure 4.22(b) shows a trace generated by Uppaal in which the claim (5) performing the data transport property fails in the presence of a black hole attack. The black hole node A behaves normally when forwarding the RDP (msg02) and REP (msg03). But when the node S transmits the data (msg04), the black hole node does not forward it. Note that the RDP message (msg07) from the node Na is ignored by the node B as it accepts the first arriving message i.e. via the black hole attacker.

4.11 Summary

This chapter presents the results obtained after rigorous analysis of wireless routing protocols by using the formal framework. Any given protocol is checked in all possible topologies of networks of up to 5 nodes. The framework is utilized to check routing protocols in 3 categories: (i) routing protocols not intended to be secure (TinyOS Beaconing, MCF, LEACH, Rumour Routing, Directed Diffusion etc) (ii) secured protocols whose vulnerability to the DoS attacks has already been identified by researchers (Authentic TinyOS with μTESLA) and (iii) protocols currently considered secure against DoS attacks with no weakness identified (ARAN, Arrive and INSENS). The model-checker generates a trace automatically confirming how an attack is possible in a particular topology for all these routing protocols. Computer simulation were used, where appropriate, to augment the results of formal modelling. This research has demonstrated that the combination of formal modelling, model-checking and computer simulation is capable of generating new results in the analysis of DoS vulnerabilities in wireless routing protocols. The approach has been used also to assist in the development of a new protocol that can be shown to be more resistant to DoS attacks. This work is described later in the dissertation.

Chapter 5

A Proposed New Protocol RAEED: Design and Evaluation

5.1 Introduction

In Chapter 2 some published secure routing protocols have been briefly discussed. It was obvious that most of the protocols are designed to address a single DoS attack and still have some drawbacks. Chapter 4 further confirmed that some well known secure routing protocols still fail against DoS attacks when they are tested rigorously using a formal modelling. It therefore leads to development of a new routing protocol that should pacify most of the attacks and to be verified against all possible executions using formal modelling. This Chapter describes the design of the new protocol in detail. We named this new protocol RAEED (Robust formally Analysed protocol for wirEless sEnsor networks Deployment). An innovative design method of using the combination of formal modelling and simulation is presented in this chapter. RAEED has been tested under different conditions like noise, varying node densities, different network sizes (scalability) etc. RAEED gives high throughput even by using a single path, has a low message overhead, does not require any special hardware and is suitable for resource constraint WSN nodes. For simplification RAEED is divided into 3 main phases: Key Setup Phase (KSP); Route Setup Phase (RSP) and Data Forwarding Phase (DFP). The KSP involves exchange of keys between nodes. In the RSP, nodes exchange the route information between themselves e.g. neighbour node IDs and hop distance from the BS. Finally the DFP involves data forwarding towards the BS. Different options have been discussed in each phase. Moreover, each phase is checked independently using formal modelling and computer simulation. The only unique thing in this protocol is that it requires nodes to transmit some messages using high power, a feature available in most WSN nodes. The protocol assumes an encryption scheme already present and the information regarding cryptography is exchanged in the KSP. The only overhead in RAEED is that, apart from neighbour information, a small data base needs to be stored in each node. However, the size of that data base is not so large that it cannot fit into the limited

memory space of WSN.

The chapter is organized as follows: the shortcomings in the previous secure protocols is briefly discussed in Section 5.2; a brief overview about RAEED is presented in Section 5.3, the three phases (KSP, RSP and DFP) of RAEED are explained in Sections 5.5, 5.6 and 5.7 respectively. Finally, a summary of the chapter is discussed in Sections 5.8.

5.2 Evaluation of Secure Routing Protocols

The current thesis has divided secure routing protocols into classes in Section 2.5 namely Multiple-paths schemes, Probabilistic path selection schemes, schemes that overhear a neighbour communication, schemes using use specialized hardware, schemes using topology mapping and cryptographic schemes. In WSNs the usage of *extra hardware* and *topology mapping* are discouraged because the former will increase the cost of nodes and the later will increase communication overheads. Normally the use of built-in hardware is preferred in WSN. Topology mapping periodically consumes a significant bandwidth and node energy. Due to their associated problems, both mechanisms, are not feasible for large networks. Thus, these solutions are mostly adopted in ad-hoc networks and rarely in small sized WSN networks. A notable example is INSENS [17, 18] which, after careful consideration, developers replaced a topology mapping technique by a modified version [19]. It is worth noting here that most of the solutions that use extra hardware are presented for ad-hoc networks rather than WSNs acknowledging the constraints of WSN.

Most routing protocols rely on *cryptographic techniques* to protect against WSN attacks in the belief that encryption solves all kinds of attacks. These protocols require configuring encryption keys and the creation of a centralized/distributed key repository. WSN, having limited resources, lack the computational power to support such encryption overheads. Some secure routing protocols do make use of efficient SKC; however these have other extraneous requirements (clock synchronization or geo-positional systems). These solutions thus might be possible for an ad-hoc networks where nodes may have high resources but not for the WSN nodes. It has been proved that the traditional encryption mechanism is not feasible in the WSN (high storage and bandwidth overhead). Moreover, the modified versions of cryptography used in the WSN cannot achieve the desired security requirements. Work on the EYES Project [232] claims that a brute force cryptographic framework is incapable of solving all the problems. Due to the lack of infrastructure in WSNs, the attacks like wormhole, INA, hello flood, rushing and jamming etc. are still possible in the presence of secure encryption schemes. On the other hand all the routing protocols must have some kind of cryptographic mechanism to get at least a secure data transfer. This is because the attacks like spoofing, sybil, etc can only be avoided if encryption remains intact.

Probabilistic path schemes can delay the attacker's job but cannot pacify attacks. Moreover,

the data is always routed towards the BS, so the attacker can launch attacks on nodes near the BS, as is the case with *multiple paths*. It enables protocols to be more robust against attacks at the expense of extra energy consumption. But removal of attacks altogether is not guaranteed. An attacker might be present in all the paths. Again, if an attacker is near to a BS it can control all data traffic. Thus the belief that multiple paths will eventually prevent the DoS attacks is also not true. A centralized approach (using BS) is not economical in WSN, so the nodes themselves must locally employ some neighbourhood watch approach locally.

The *neighbourhood watch schemes* are sometimes vulnerable to the framing attack and high error rate due to collisions, unidirectional links and the weak signals. But if the nodes only monitor themselves and decide to forward data depending on neighbour's performance, the framing attack threat will vanish straight away. The collisions will occasionally induce some errors enabling the nodes to exclude neighbours that have a weak transmission signal. As the bidirectional links are a must for neighbourhood watch, the neighbours that have unidirectional links should also be removed.

Finally, most of the protocols only address one or a few attacks but not all. A user must be given a solution that prevents most of the attacks and not just one or two. It is infeasible to use a combination of different protocols to address all attacks. Even if it has been feasible this combination will add more burdens on the WSN resources and available bandwidth. The current research is of the view that there is a serious need for developing a single protocol that, in some way, should addresses most of the attacks. Also it should not require considerable additional resources so that it can fit into the WSN environment. The protocol may not avoid all attacks but should be workable in the presence of most of the attacks. Further other factors should be considered such as cost, limited resources, low overhead traffic, high density and scalability. The enhanced version of INSENS does fulfill some of these requirements. It is claimed to be a solution against number of DoS attacks like black hole, gray hole, hello flood, rushing, jamming, spoofing, wormhole and sinkhole. The encryption is provided using OHC, the robustness is provided by the multi paths to multiple base stations and the new INSENS version avoids topology mapping thus providing scalability. However, the formal framework confirmed (Chapter 4) that the INSENS protocol only pacifies hello flood and spoofing attacks; while other attacks are still successful (Section 6.7.2.1).

5.3 The New Protocol: An Overview

By analyzing several published routing protocols, both secure and insecure, it was evident that there remains a need for a new routing protocol that is more robust than the existing routing protocols. Formal analysis of one of the most robust protocols, INSENS, confirms that there is still room for improvement in it. Both the INSENS and the LEAP protocols use bidirectional verification techniques which, the formal framework had confirmed, can avoid

the hello flood attack. Moreover one way hash chain, used by both the protocols, can provide authentication and confidentiality. So instead of starting from the scratch, it was decided to employ the bidirectional verification and the key exchange characteristics of the INSENS and the LEAP. The bidirectional verification method was improved so that it preserves the security with reduced traffic. Note that the research is not concerned with encryption weakness in both schemes and it is rather concerned on issues related to the DoS attacks. The encryption issues are left as future work and any cryptography scheme can later replace the one presented in the research to improve encryption. Therefore, this research only proves (Section 6.2), by using formal modelling, computer simulation and practical implementation, that RAEED is immune from the hello flood attack.

The path/route propagation of the RAEED involves the time stamps propagated by the BS, that assigns each node an authentic level or hop count distance from the BS using one-way hash chains (OHC). It is believed that if the path is accumulated in route update phase, e.g. in protocols like INSENS etc, a high overhead is added on message size especially in the large networks. The path accumulation is thus avoided to overcome this overhead. Moreover, almost all the protocols are vulnerable to the INA and the wormhole attack in the route propagation. Therefore, to avoid these attacks a new methodology has been presented in Section 6.3 that employs transmission power to avoid these attacks. Again the encryption issue related to OHC are simulated rather than the actual implementation of OHC schemes. However the one way property of OHC is preserved when implementing attackers.

Data forwarding to the parents and the use of multiple paths is also avoided in RAEED. The formal framework and simulation have confirmed that, in several routing protocols, data may not reach the destination even by employing the multiple paths especially in low density networks. An innovative technique is used for data propagation that depends on the neighbour's performance, (evaluated at runtime by each node). Thus the main aim of different attacks (wormhole, INA, black hole, jamming etc) to prevent data from reaching the destination is pacified through this technique. The current research calls this technique *Neighbourhood Watch*. It neither requires any special hardware (GPS, Directional Antennas etc), nor highly resource nodes (like Guards [99] etc). Instead, by employing the resource constraint nodes, the attacks are pacified by finding an alternative path. Thus data will eventually be received at the BS. Finally, the protocol does not require multi-path routing; it provides high throughput even by employing a single path. This makes the RAEED useful in terms of low bandwidth overheads and avoids employing multiple BSs.

The only overhead with RAEED is the maintenance of a small data base (DB). This database is a required as each node has to store the neighbour's performance, the data payload history, and the events/source information. An important table to point out is the Neighbour Table (NT). NT contains all the information about the neighbour nodes. This DB is discussed in detail in Section A.7.

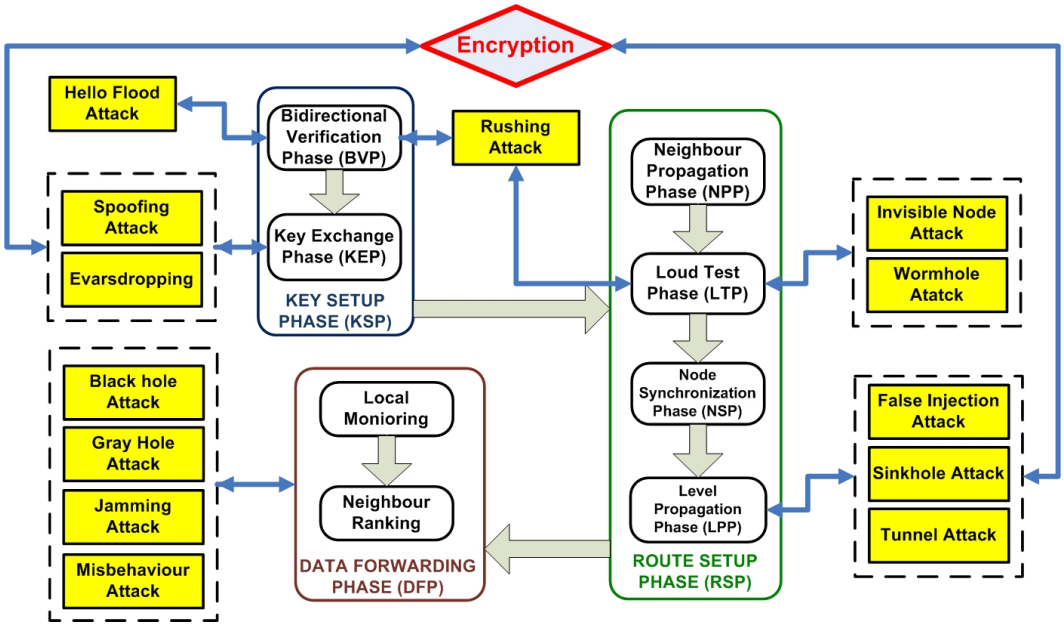


Figure 5.1: RAEED Protocol Phases

The complete protocol design can be divided into 3 main phases namely Key Setup Phase (KSP), Route Setup Phase (RSP) and Data Forward Phase (DFP). Each phase is further subdivided into different phases, the aim of each is to pacify a particular attack. A complete block diagram of all the phases and the attacks that are rectified is shown in Figure 5.1. Note that some attacks rectification is dependent on cryptography and the protocol requires encryption to be successful for rectification of those attacks. As previously stated this research is not concerned with the cryptography; it is assumed that an encryption mechanism is already present. The 3 phases are briefly as follows:

Key Setup Phase (KSP) is subdivided into *Bidirectional Verification Phase (BVP)* and *Key Exchange Phase* (KEP). The BVP involves broadcasting a message and expecting a reply in response to confirm that the links are bidirectional. The BVP thus resists the unidirectional links and the hello flood attackers. The KEP involves the exchange of keys used in cryptography. The message in the next phases are encrypted with those keys to pacify the attacks like spoofing and eavesdropping. The encryption involved in the current protocol piggybacks keys in the BVP and thus skips the KEP. But if a different encryption mechanism is used the keys can be exchanged in the KEP. Note that as the key mechanism of the INSENS/LEAP is adopted, the protocol is vulnerable to the shortcomings associated with them like the global key revocation. Thus the protocol should finish KEP very quickly if the current scheme is used. As stated earlier the improvement in KEP is left for future work.

Route Setup Phase (RSP) is subdivided into *Neighbour Propagation Phase (NPP)*, *Loud Test Phase (LTP)*, *Node Synchronization Phase (NSP)* and *Level Propagation Phase (LPP)*.

NPP involves exchange of verified (authentic if encryption is strong) node IDs of neighbours detected in the KSP. All the nodes of a network will have knowledge of 2-hop neighbours after the NPP. Nodes already know their 1-hop neighbours after the BVP of KSP. The *LTP* involves loud beacons exchanged between 2-hop neighbours to discover virtual links between nodes (INA and wormhole attack). The NPP enables all nodes to roughly synchronize with the BS time. Finally, the LTP involves assigning each node hop distance from BS in an authentic manner. The nodes also discover their neighbour's hop count distance in the LTP. The LTP, along with an encryption technique (one way hash chain), can solve attacks like sinkhole and the false injection.

Data Forward Phase (DFP) or Operation phase involves local monitoring and neighbour ranking based on the data forwarding performance of the neighbours in order to solve attacks like the black hole, gray hole and jamming. This phase also enables the protocol to pacify the misbehavior (selfishness) and the framing attacks.

5.4 Message Format

The protocol follows a specific message format for all the three phases. The format of message used in RAEED is:
$Type, SenderID, TargetID, Nonce, PairKey, ClusterKey, MAC, OHC, Payload$
These fields are defined as:

- The *Type* is the 8-bit message type or header.
- *SenderID* and *TargetID* are the 16-bit IDs of Sender and Target nodes.
- *Nonce* is an 8-bit nonce attached to a message in the *Key Setup Phase* (KSP) and later it is the time stamp attached by a BS in the *Route Setup Phase* (RSP).
- The *PairKey* (PK) and *ClusterKey* (CK) are the 8-bit keys used in KSP. Note that the cluster key is piggybacked along with the pair key in the KSP. Depending upon encryption and size of the keys, one can increase the sizes of Nonce, Pair Key and Cluster Key if so required. In DFP the PK and CK are combined to address a 16 bit Event ID.
- The *MAC* is the 8-bit Message Authentication Code attached to the message by the BS using a one-way hash chain (OHC). OHC works in one way so a node can decrypt to confirm that the hash is correct but cannot generate a new hash. Depending upon the size of the MAC this field can be increased in length.
- The *OHC* (8-bits) contains the next chain value of the hash function and is updated by the BS. Combined with the MAC it forms a 16 bit Source ID in data forwarding. This field can be increased in length depending upon size of the OHC.

- *Payload* is reserved for the data payload

5.5 Key Setup Phase (KSP)

This research assumes that there is a cryptographic scheme already present in the protocol. In the Key Setup Phase (KSP) this research explains how to employ the cryptographic technique in RAEED. This also enables key exchanges between the nodes. Moreover, the protocol also requires the bidirectional verification, as it has been proved by the formal framework in Chapter 4 that the bidirectional verification will prevent the hello flood attacker from launching an attack. The KSP addresses all of these issues. The schemes that adopt bidirectional verification are LEAP and its extension INSENS. RAEED is a modified version of these two schemes.

This section is further categorized as: Section 5.5.1 explains the design of KSP in a semi formal manner using equations and message sequence diagrams; Section 5.5.2 describes formal modelling to confirm that the KSP provides the bidirectional property; Section 5.5.3 uses computer simulation to confirms that the KSP presented has low message overhead and thus high efficiency; Section 5.5.4 employs the formal modelling and computer simulation to check the effect of the noise and the collisions on KSP; Section 5.5.5 involves confirming that the KSP provides the same security properties as INSENS by employing formal modelling; while further simulation results on the KSP e.g. examines the effect of different factors (different delays and redundancy employed in the low level software) are described in Section 5.5.6.

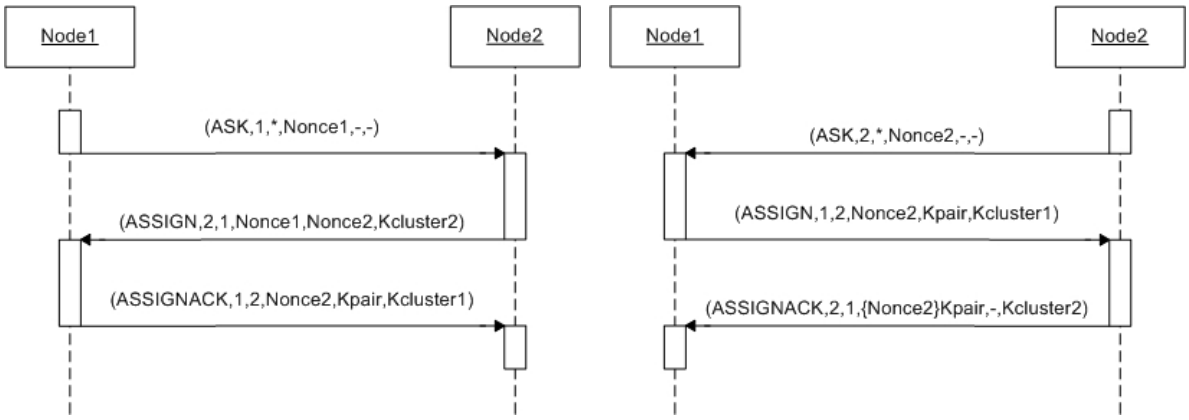
5.5.1 The Design

When a node is booted it tries to verify its neighbours by employing bidirectional verification. The keys are also exchanged during this phase. The principle of using 4 different keys suggested by LEAP and used by INSENS is adopted. The fields of a message reserved for KSP are Type, SenderID, TargetID, Nonce, PairKey, ClusterKey. All the fields are self explanatory with fields reserved for IDs of sender and the target nodes, nonce, a random number generated for cryptography, the pair-key used between a pair of neighbour nodes; and cluster key employed by a group of neighbours.

It is assumed that all the nodes are preloaded with a common global key, K_G. This is the principle adopted by both LEAP [43] and INSENS [19] and it is assumed that the key is deleted after a few seconds of deployment. After a node is booted, it broadcasts an ASK beacon containing a random nonce and its own ID encrypted using the global key, K_G, known to all nodes:

$$N \rightarrow * : (ASK, [ID_N, -, nonce, -, -]_{K_G})$$

The receiving node, R, updates its NT by adding this new neighbour N. R also randomly generates a pair-key and a cluster key, if these keys are nonexistent for the neighbour N. The



(a) Case 1 (b) Case 2

Figure 5.2: Two cases in initiating the Bidirectional Verification Phase

ASK message is then replied with an ASSIGN beacon which contains the received nonce, pair key $K_{R,N}$ and a cluster Key K_C. The message is encrypted with the global key, K_G:

$$R \rightarrow N : (ASSIGN, [ID_R, ID_N, nonce, K_{R,N}, K_C]_{K_G})$$

If both nodes generate separate pair keys then each node will have two pair keys instead of one for each neighbour. In order to resolve this issue, the node with lower ID generates a key while the node with higher ID sends a junk number instead to fulfill the bidirectional verification process. Note that unlike INSENS, and like LEAP, RAEED unicasts the cluster key in the bidirectional verification phase. The receiving node then sends an ASSIGNACK beacon to complete the bidirectional verification at both ends. The ASSIGNACK contains a nonce and the keys similar to the ASSIGN beacon:

$$N \rightarrow R : (ASSIGNACK, [ID_N, ID_R, nonce, K_{R,N}, K_C]_{K_G})$$

In all the circumstances, there will be two cases: (i) either the node with higher ID initiates ASK first or (ii) the node with the lower ID initiates it first. Both cases are addressed in Figure 5.2(a) and Figure 5.2(b) respectively. The figure shows communication between the two nodes having IDs 1 and 2 for both cases. In the case, where Node1 has initiated ASK first, it broadcasts its nonce with other fields being null. The Node2 in return replies with the ASSIGN, containing the nonce of Node1 instead of the pair-key and its cluster key. Note that as Node2 has the higher ID it has no right to send its pair-key. Instead it attaches its own nonce that will be used in bidirectional verification from Node1 to Node2 and the cluster key. The Node1, on receiving the ASSIGN message, adds Node2 as a verified neighbour. It then responses back with the ASSIGNACK. However, at this time it transmits the nonce of Node2, the pair-key (it has lower ID) and its cluster key. When Node2 receives this message, it also adds Node1 as its verified neighbour and updates the keys. All the messages having been encrypted using

the global key known to all nodes. The Node2, on receiving the ASSIGNACK message, adds Node1 as a verified neighbour.

In the second case, when Node2 initiates the ASK beacon by sending its own nonce, Node1 replies back with the Node2's nonce and the newly generated pair-key because Node1 has lower ID. On receipt of the ASSIGN beacon, Node2 adds Node1 as the verified neighbour. It then responds back with its nonce encrypted with the pair-key sent by Node1 instead of the global key. Note that Node1 has not sent the nonce in this case. So the verification process can only be done if the pair-key sent by Node1 is used. On receipt of the ASSIGNACK, Node1 decrypts the nonce and if correct, it adds Node2 as its verified neighbour.

In noisy conditions or because of collisions, some messages are likely to be lost. So the ASK beacon is sent after 3 random times intervals. A node, upon receiving the ASK beacon, only responds back with the ASSIGN beacon if it has not received the ASSIGNACK beacon from the same node. Thus redundant messages are avoided. The overhead here is the extra 2 ASK beacons per node. A timer is fired in a node, after the KSP has finished. This enables the unicast of a redundant ASSIGN beacon to all the unverified nodes (verified flag is clear in NT). An entry in NT indicates that the neighbour has sent an ASK beacon but hasn't replied with an ASSIGNACK when it unicasted the ASSIGN beacon. This may result from either the non-malicious (collision/noise) or the malicious (DoS attack) reasons. This attempt should be made a limited number of times say n, to limit the depletion of energy of the nodes caused by a hello flood attacker.

The KSP in RAEED is different from LEAP and INSENS. A node, after receiving an ASSIGNACK from a particular node, will not send any ASSIGN beacon to that node again. The reception of ASSIGNACK is a guarantee that the node has already received an ASSIGN beacon along with the keys. Thus, further ASSIGN beacons are not required; this reduces the message traffic in this KSP. In INSENS and LEAP, each node broadcasts an initial beacon, and then unicasts a reply & cluster key beacon. Table 5.1 compares the number of messages transmitted in the KSP for LEAP, INSENS and RAEED in an N-node fully connected network:

Table 5.1: Table comparing number of messages in KSP of RAEED,LEAP and INSENS

Protocol	**First message**	**Second message**	**Third message**	**Total messages**
INSENS	N echo	N(N-1) echoback	N(N-1) CLUSTER	N(2N-1)
LEAP	N HELLO	N(N-1) ACK	N(N-1) ACKBACK	N(2N-1)
New protocol	N ASK	$\sum_1^{N-1}$ ASSIGN	$\sum_1^{N-1}$ ASSIGNACK	N + 2 $\sum_1^{N-1}$

Looking at Table 5.1, it is clear that the RAEED requires fewer message exchanges compared with INSENS/LEAP in any N node network with N-1 neighbours. Thus RAEED is an efficient

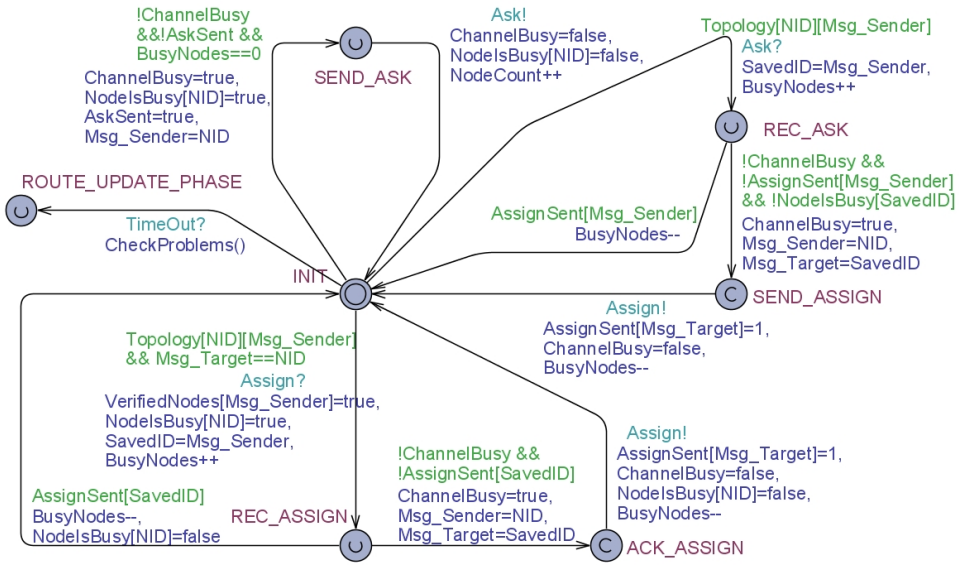


Figure 5.3: Uppaal model for improved Key Setup Phase in RAEED

form of KSP of INSENS and LEAP because it piggybacks cluster key along with the pair-key exchange. This point will later be demonstrated using simulation (Section 5.5.3).

5.5.2 Formal Verification of Bidirectional Property

In this section formal verification is used to test the following hypothesis:

"The modified Key Setup Phase of RAEED will provide the same bidirectional verification properties as provide by INSENS and LEAP"

The assumptions stated in Section 4.3.3 are persisted in this model and no attack is considered in this model.

5.5.2.1 Formal Model

RAEED is different from LEAP/INSENS so a formal model in Uppaal is developed to verify if it still fulfils the required bidirectional properties. The formal framework presented earlier in Section 4.3 is used for evaluation of RAEED in this chapter. The model to confirm bidirectional property is composed of two parts a node model and an event generator model. The ***node model*** is shown in Figure 5.3. It starts from the INIT location and tries to broadcast ASK beacon after finding that the channel is free by going to the SEND_ASK location. In case a node receives an ASK beacon from a neighbour it moves to the REC_ASK location. If the node has already sent ASSIGN (checked using the `AssignSent` flag) then it will go back to the INIT location, otherwise it will move to the SEND_ASSIGN location, transmits the ASSIGN beacon and go back to the INIT location. Upon receiving an ASSIGN beacon, the model moves to the REC_ASSIGN location, sets the *`VerifiedNodes`* flag and then goes to the ACK_ASSIGN

location. It then transmits the ASSIGN beacon and moves back to the INIT location again. The ***event generator model*** has 2 locations. It starts in the KEY_SETUP_PHASE and then moves to the ROUTE_SETUP_PHASE when all nodes finish the KSP.

5.5.2.2 Verification

The claims verified were:

1. All nodes will finish the KSP

2. When the KSP finishes, then all the nodes have correct verified neighbours

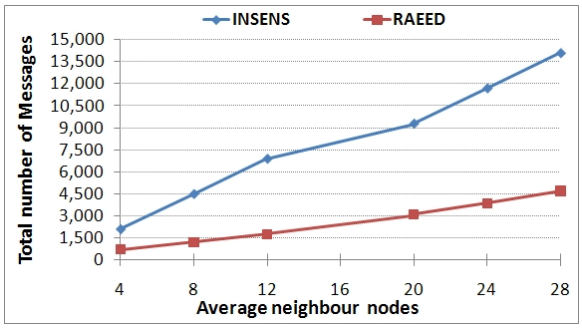
A detailed description of the above claims in terms of Uppaal properties is explained in Section B.2.1.

Both the claims were proved confirming the desired behaviour hold true in RAEED.

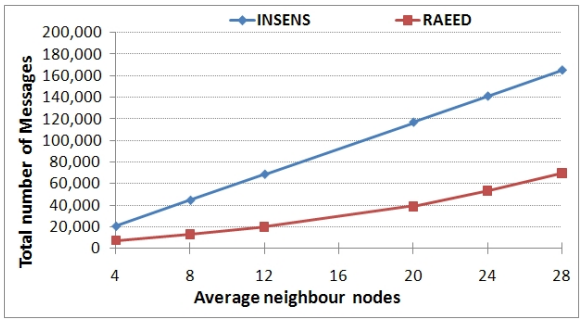
5.5.3 Simulation Results to Evaluate Efficiency

It has been stated earlier that RAEED reduces the number of messages exchanged between nodes. In this section the computer simulation has been employed to perform this evaluation in terms of scalability. To confirm this the total number of the messages exchanged between the nodes in the KSP have been measured for different densities (nodes in range) of a 100 node network, placed in a grid placement of 10x10. A total of 20 experiments were performed for each density value and their average value was computed. For these experiments, each node broadcasts 3 ASK messages. For a valid comparison the results were calculated by modifying INSENS also to send 3 echo messages instead of 1. The results are shown in Figure 5.4(a) for RAEED along with the INSENS protocol. The average number of neighbour nodes (density) was varied between 4 and 28 (Section A.2.1.3). It is evident that the number of messages is reduced by about 30% for all cases. To confirm that the KSP fulfills its requirements, the Neighbour Table (NT) of all nodes were also checked in all the experiments to confirm that they contained the correct entries.

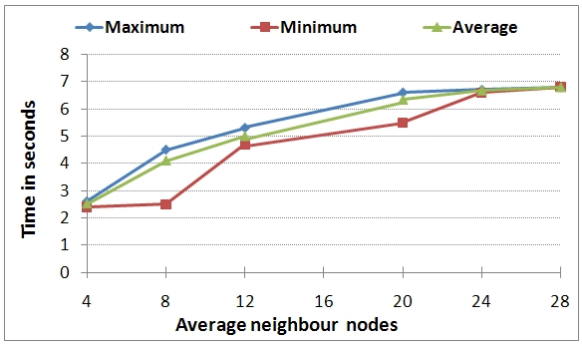
For each of the tests performed to check KSP, two problem files were generated to confirm whether the objectives of this phase were achieved. The first file contained the entries in the Neighbour Table (NT) that could not be verified during that phase. This percentage remained 0% up to 12 node density and then it is increased gradually to 0.05%, 0.11% and 0.45%. Note that even for the 28 node neighbours, the highest average density considered, the unverified legitimate neighbour percentage was very low (less than 0.5%) and so can be neglected. The reason for this problem is the collision (even after using 3 redundant messages) as the experiments were performed in ideal conditions (i.e. no noise). To solve this issue a one time redundant ASSIGN message was sent once for all unverified neighbours. If in return an ASSIGNACK was received, then that neighbour's status was altered to have been verified.



(a) Total messages exchanged in KSP in 100 nodes grid network



(b) Total messages exchanged in KSP in 1000 nodes grid network

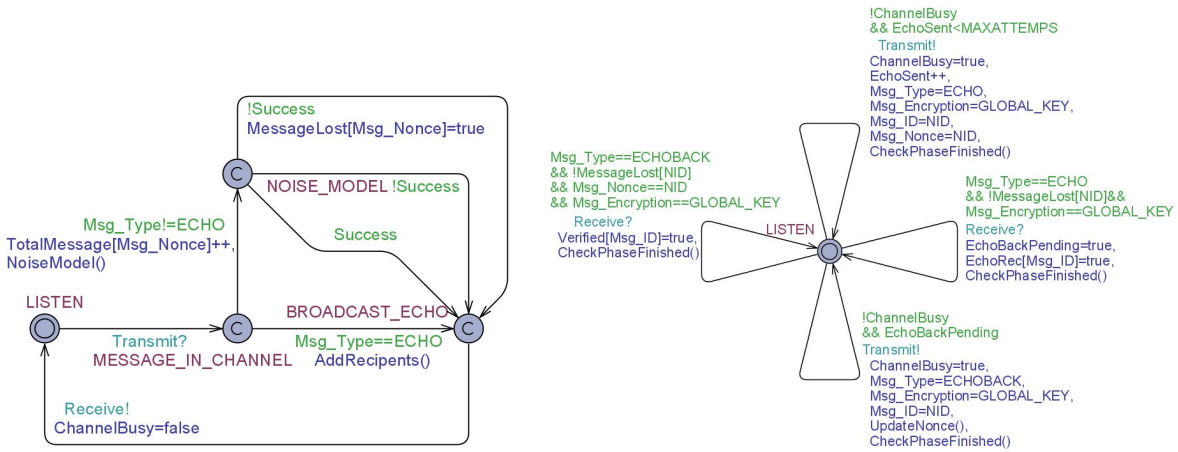


(c) Time needed to complete the KSP in 1000 node grid network

Figure 5.4: The effect of density on total messages exchanged in KSP

The second problem file recorded the number of missed neighbours. The results in this file were different from the last case mentioned. These are the neighbour entries totally missing in the NT because not all the sent ASK beacons are received as a result of collisions. In the previous case, it was the entries in NT that had not been verified. This percentage remains 0% up to 20 node densities and then it is gradually increases to 0.02% and 0.68% for 24 and 28 average neighbours respectively. This was expected because of the large number of nodes within each other's radio range; which increases collisions and some neighbours may have been missed. The solution to this problem is to increase the number of ASK beacons from 3. However, as the error percentage is quite low so this problem can be ignored.

The experiments were then repeated on RAEED for a larger network of 1000 nodes. The results are shown in Figure 5.4(b). Note that the density was also increased up to 28 average neighbour nodes. The results again confirmed that even at high density and high number of nodes RAEED resulted in about one third of the message overhead compared to INSENS. The



(a) Channel model

(b) Node model

Figure 5.5: Uppaal model to check effect of message loss on INSENS with separate channel model

error percentage in NT was also checked for 1000 nodes. This time, as the density was high, so both values increased as expected. The entries in the NT that cannot be verified during this phase remained 0% up to a density equal to 12 and then increased by 0.01%, 1.8%, and 5.57% for higher densities of 20, 24 and 28 respectively. The number of missed neighbours also remained 0% up to 12 node density and then is increased to 0.1%, 0.7%, and 1.8% respectively. Note that these errors are again negligible considering the number of neighbours available to each node at higher densities.

The total time spent to complete the KSP was also studied. This is shown in Figure 5.4(c). Note that the expected time to finish for all the network topologies is less than 3.0 seconds (3 ASK beacons per second). Extra time is taken in sending the redundant beacons as a result of missed neighbours following collisions. This shows that with high density and in the presence of noise this setup time may increase. However, one can decrease this extra time by sending ASSIGN beacons at a lower interval after the expected setup time is finished (i.e. to reduce the delay between sending ASSIGN beacons later). This issue has been discussed later in Section 5.5.6.2.

5.5.4 Effect of Noise and Collision on the KSP

This section involves testing the KSP under noisy conditions, i.e. when a channel is not ideal. Formal modelling is first employed to check the effect of noise and the results are later confirmed with those obtained using computer simulation.

5.5.4.1 Formal Model

The hypothesis for this model is:

"RAEED is more robust to noise than INSENS and LEAP".

The assumptions for this modelling are different from the ones stated in Section 4.3.3. The channel is no longer ideal and data may be lost due to noise or external effect. Also it is assumed that the first message from each node (e.g. echo for INSENS) will always arrive at the destination node, i.e., it is never lost. Finally, only a single topology in which all the nodes are connected is considered.

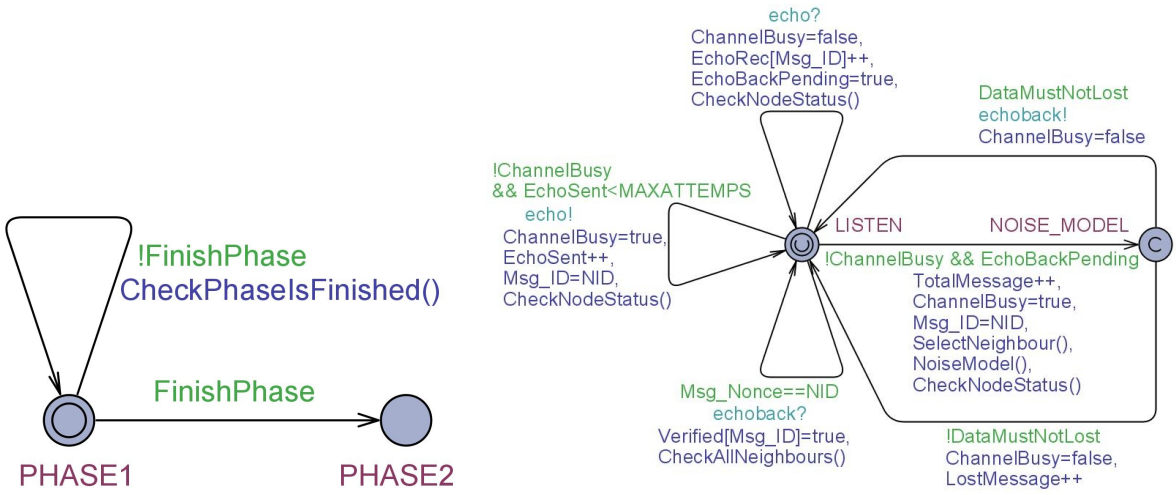
Initially the KSP part of INSENS (Phase 1) is modelled in the presence of noise. The model is divided into 3 parts the node, the event generator and the channel. Note that the channel is separately modelled to introduce message loss instead of the normal reception of messages.

The ***node*** model (Figure 5.5(b)) has only one location (LISTEN) and 4 self transitions. Each self transitions depends on what message is received or transmitted from/to channel and it is either echo or echoback. An echo can be sent at the most `MAXATTEMPTS` times. Upon receiving an echo, the `EchoBackPending` flag is set along with the corresponding node's `EchoRec[]` flag to indicate which neighbour should be sent an echoback. The `EchoBackPending` flag enables sending of the echoback message to the channel and a function `UpdateNonce()` finds the node to which the message should be sent (ID). For simplicity, the nonce is modelled using the node ID, i.e. each node sends its ID in the nonce field instead of a random number. This is done to save state space so that each node does not have to store its nonce separately along with the nonce of each neighbour. Finally, upon receiving the echoback, a node sets the corresponding `Verified[]` flag. Note that all messages are encrypted with the global key. Also, all the transitions execute the `CheckPhaseFinish()` function to check if the node has finished its Phase 1. This will set the corresponding global boolean array `FinishNode[ID]`.

The **channel** model (Figure 5.5(a)) starts in the LISTEN location. If any node transmits a message a global variable `ChannelBusy` is set and model goes to the MESSAGE_IN_CHANNEL location. If the message type is echo, the corresponding recipients are added to the message using the `AddRecipents()` function. The model then moves to BROADCAST_MESSAGE, which is a critical location. Thus the message is broadcast to recipients and `ChannelBusy` is cleared enabling other nodes to transmit a message. In case the message is not echo, the function `NoiseModel()` checks if the messages received by the receiver are below a threshold (e.g. 70%) or not. In case a message is below that level, it will always be transmitted to that node (`Success` is true) otherwise the message may or may not be received by channel (nondeterministic manner). The Channel model always updates the number of messages received and lost by all nodes to calculate the threshold stated earlier.

The ***event generator*** model has 2 locations PHASE1 and PHASE2. The model remains in the PHASE1 until all node models moves out of the PHASE1 location. A self transition checks status of the node models using function `CheckNodesState()`.

Although the model yields the desired results when simulated, due to the model complexity, the state space explosion problem hinders the verification process. A simple sanity check that



(a) Event Generator model

(b) Node model

Figure 5.6: Uppaal model to check effect of message loss on INSENS without separate channel model

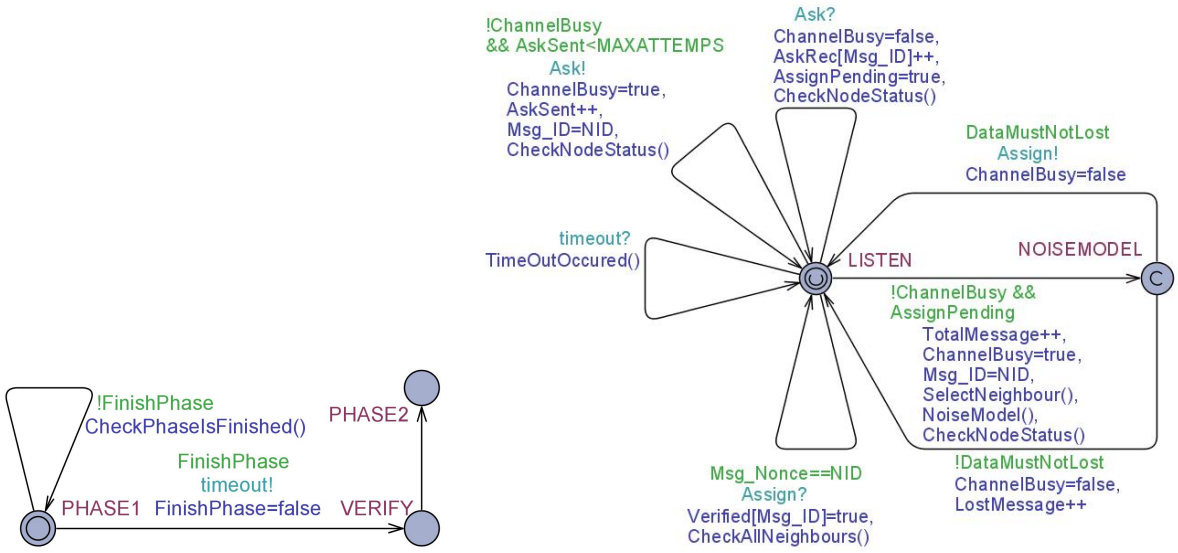
the event generator eventually goes to PHASE2 takes more than an hour to verify even with 3 nodes network:

$$E <> Protocol.PHASE2$$

The introduction of the channel model induced the state space explosion problem, also experienced while modelling with Spin. Note that the Spin models also employed a separate channel model. So a simplified model of INSENS was developed to accommodate message loss inside the node model instead of the separate channel model. All unnecessary details like encryption etc are also removed as the aim is to check effect of noise. The modified ***node*** model is shown in the Figure 5.6(b). It is almost the same as in Figure 5.5(b). Note that the echoback might be lost if the number of messages lost (`MessageLost`) is below a threshold (15% set for model). The threshold is calculated by the model before transmitting by:

$$Threshold = 100 \times MessageLost / TotalMessage$$

The function `NoiseModel()` calculates the threshold and then determines if the message must be forwarded or lost (to simulate noise). If the noise is below the threshold it will always be received by a neighbouring nodes (`DataMustNotLost`). The function `CheckNodeStatus()` is available in most transitions and confirms if the current node has get out of the PHASE1 location i.e. it has sent echo and no echoback is pending to be sent to any neighbour. A flag `NodeFinish[NID]` is set or cleared based on the outcome of these results. When all nodes have move out of the PHASE1 location, the protocol moves to the PHASE1 location as shown in Figure 5.6(a). On receipt of echoback, the corresponding `Verified[NID]` flag is set and a function `CheckAllNeighbours()` updates the problems left in the NT of that node. Note that



(a) Event Generator model

(b) Node model

Figure 5.7: UPPAAL model to check effect of message loss on RAEED

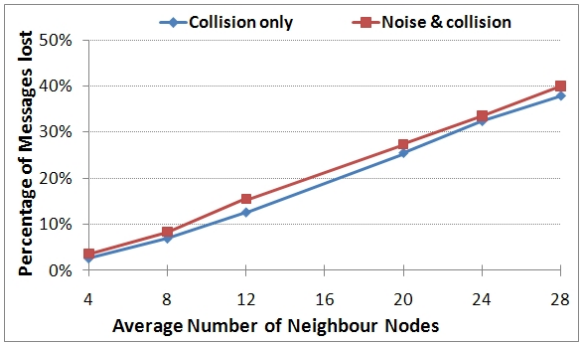
any remaining unverified node will update the `Problems` variable. A flag `EchoBackPending` is set when an echo is received as well as the corresponding `EchoRec[]` variable for a neighbour. A function `SelectNeighbour()` is used to update the message nonce before sending echoback which will depends on `EchoRec` variable.

The claims/properties verified in this model were:

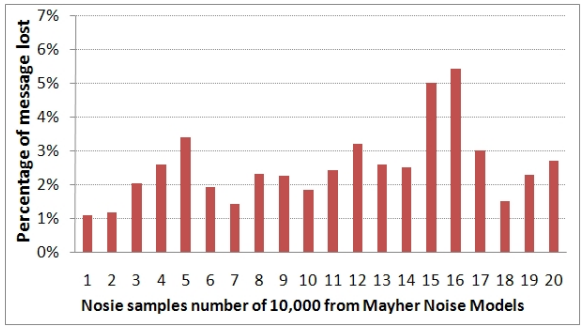
1. All nodes eventually enter Phase 2
2. All 3 echo messages can be sent by each node
3. When nodes enter Phase 2 there is no unidirectional link in the NT

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.2.2. The liveness property (Claim 3) however fails for INSENS even with a 10% message loss threshold. The formal model for RAEED was then implemented in Uppaal to accommodate the noise, as it was done for the INSENS protocol. The new RAEED model is composed of an event generator and a node models. It is evident that an extra location VERIFY, has been added in the ***event generator*** model (Figures 5.7(a)) for RAEED comparing with that of the INSENS model. This is done because RAEED, after finishing Phase 1, and before going to Phase 2, checks the unverified neighbours by making one further attempt at rectifying unverified neighbours. The VERIFY location triggers a timeout in all the nodes. The nodes, upon receiving a timeout, call the `TimeOutOccured()` function, in which nodes try to re-establish links with unverified neighbours by sending an extra ASK beacon.

The ***node*** model for RAEED (Figure 5.7(b)) has 2 locations NOISE_MODEL and LISTEN. This model is similar to the node model presented earlier for INSENS protocol (Figure 5.6(b)).



(a) No Noise and Complete Meyer Noise Model



(b) Different samples of 10,000 from Node Meyer Noise Model

Figure 5.8: The percentage of messages lost in the KSP in 100 node grid network

The RAEED node model starts in the LISTEN location and the self transitions update the model upon receiving Ask/Assign message or a timeout. When an Ask message is received, the model sets the `AssignPending` flag, which indicates that the model must send an Assign message in reply. The functions `CheckNodeStatus()` and `CheckAllNeighbours()` are already explained in the INSENS node model. Upon receiving an Assign message, the `Verified` flag is set indicating the sender is added as a verified neighbour. The model keeps on transmitting the Ask messages until they have been sent `MAXATTEPMTS` times. The variable `AskSent` counts this parameter. If the channel is free and an Assign message is pending (`AssignPending` flag is set), the node model moves to NOISEMODEL location. The message is then sent by employing the model as explained earlier in Figure 5.6(b).

It has been confirmed that now the liveness property (Claim 3) is satisfied and only fails if the noise threshold is increased to 21%. Below this value this property is satisfied, whereas in INSENS's case this property fails even if the threshold value is 10%. Note that by adding more redundancy the threshold value can be increased above 21%.

The INSENS and RAEED models confirmed that a loss of 10% of messages (noise) will introduce errors in NT. It is evident that even a 3 node network may contain wrong entries in NT. Note that even in the absence of noise, collisions may enable message loss in the practical systems. So some redundancy is necessary to improve robustness in presence of noise.

5.5.4.2 Simulation Results

This section focuses on applying computer simulation to quantify the number of messages lost due to collision and noise. The experiments were performed 20 times and average values noted. In this research, the average values are mostly plotted when the variation between experimental values is less. The first set of experiments involved testing a 100 node grid network with various

densities (i.e. average numbers of neighbours); these results are shown in Figure 5.8(a) and Figure 5.8(b) respectively. It is evident that in high density networks, the number of messages lost due to collision is very high. Message collision are quite low when the average number of neighbours are less 12 (i.e. 12% maximum) but it increases as the density is increased. The message lost is about 38% when the average number of neighbours is 28. This is quite logical as more nodes within the range will always cause more collisions. It is also evident that the effect of noise remains low throughout (1-3%). Note that the increase in the number of nodes should not affect the message lost due to the noise as noise is an external effect and is independent of the number of nodes in range.

As indicated in Section A.4.3, the current research did not intend to model noise accurately. Rather the aim is to check the overall message loss affect. That is why Section A.4.3 introduces the division of the Meyer files into different samples of 10,000 and then observes its effects on data loss. The samples are then applied to a 100 node network acting in KSP. The results are shown in Figure 5.8(b). Note that these results are symmetric when compared with to results in Figure A.2(b). Although the percentage of the messages lost is different in both cases (as noise trace for each node will be different) the over all pattern remains the same for all the samples. For further noise tests, performed later (Section 5.7.3, Section 5.5.6.3 and Section 5.5.6.4) four noise samples are employed to check noise; these are shown in Table 5.2:

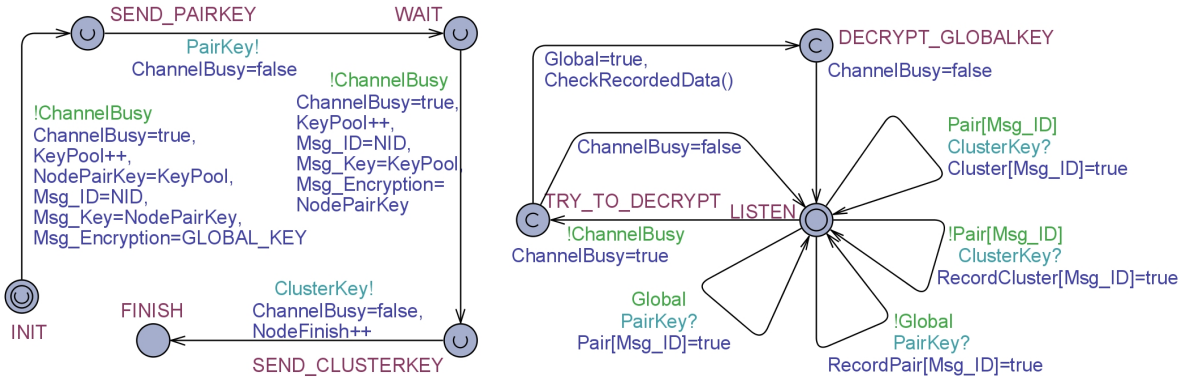
Table 5.2: Noise samples categorized by samples of 10k used to test noise

Category	**Noise file use**	**Expected message loss**
No noise	The file containing no noise	0%
Low noise	Sample 18	1.3% to 2.0%
Medium noise	Sample 05	3.0% to 4.0%
High noise	Sample 16	5.0% to 6.0%

5.5.5 Confirmation that Security Properties Hold

Although there are no major changes in RAEED as compared to LEAP and INSENS in the KSP, confirmation that security of the protocol is valid is still required. The thesis adopts two methods: visual inspection and formal modelling.

Applying ***visual inspection*** it is evident that if the global key has been captured at any stage then both the cluster key and the pair keys of a node will be revealed to the attacker, possibly by recording the messages while eavesdropping; the same is true of INSENS. The pair key is encrypted with the global key and the cluster key is encrypted with the pair key. So it is obvious that if the attacker is recording the messages, and later the global key has been revealed to the attacker, the attacker will be able to disclose the pair keys. This will also enable



(a) Node model

(b) Attacker model

Figure 5.9: Uppaal model to check security of INSENS and RAEED is same

the attacker to disclose the cluster key because it was encrypted with pair keys. So it is evident that both in INSENS and RAEED the security capabilities are the same. Both depend on hiding the global key from the attacker. Hence, decreasing the number of messages by piggy bagging the cluster keys in bidirectional verification has no effect on the security requirements. It is also now proved using the formal model.

5.5.5.1 Formal Model

The hypothesis for this model is:

"The compromise of the global key in INSENS always means an attacker has access to both the pair and the cluster key of nodes, eavesdropped earlier".

Apart from the assumptions stated in Section 4.3.3, it is assumed that an attacker can eavesdrop all the nodes in range and record all the keys whether it can decrypt them or not. Also the global key might be disclosed at any time between the start of setup phase and its completion.

The formal model is composed of 3 parts namely an event generator, the node and an attacker. The ***event generator*** model has 2 locations making the start and the end of the phase. The phase ends if all the nodes of the network enter the FINISH location. A global variable `NodeFinish` is incremented each time a node enters the FINISH location. The ***node*** model is shown in Figure 5.9(a). It begins in the INIT location. If it finds the channel to be free, it broadcasts the pair key (SEND_PAIRKEY), waits for some time (WAIT), and then unicasts the cluster key (SEND_CLUSTERKEY). Note that the channel becomes free when the node is in the WAIT location allowing other nodes to broadcast their messages. These three locations model the two phases and delay between them in real world. Once the process is complete the node model moves to the FINISH location by incrementing the global variable `NodeFinish`

(indicates how many nodes have finished the phase) and then stays in this location.

The ***attacker*** model (Figure 5.9(b)) starts in the LISTEN location and eavesdrops all messages of the legitimate nodes. Note that the attacker model tries to decrypt the global key (TRY_TO_DECRYPT) when channel is free and no node is transmitting any data as otherwise it must record that message. This attempt is nondeterministic. The attacker may (DECRYPT_GLOBALKEY) or may not decrypt the key. When the attacker model receives a pair key and it possesses the global key at that moment, it will be able to decrypt the pair key. Non availability of the pair key will enable the attacker to record the pair key for later use. Similarly, if the attacker possess the pair key when the cluster key is sent by a node, it can decrypt the key otherwise it records that key. Note that whenever the attacker is able to access the global key (DECRYPT_GLOBALKEY) the function `CheckRecordedData()` executed to decrypt all the stored data.

5.5.5.2 Verification

The claims verified were:

1. All nodes eventually enter Phase 2 (RSP)
2. All 3 echo messages can be sent by each node
3. Global key compromise does not allow the attacker to capture the other keys
4. Global key compromise leads to all further pair keys being captured
5. Global key compromise leads to all further cluster keys being captured

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.2.3. All the above claims/properties were proved confirming both INSENS and RAEED possesses same security properties in terms of node compromise.

5.5.6 Simulation Results: Effect of Different Factors on KSP

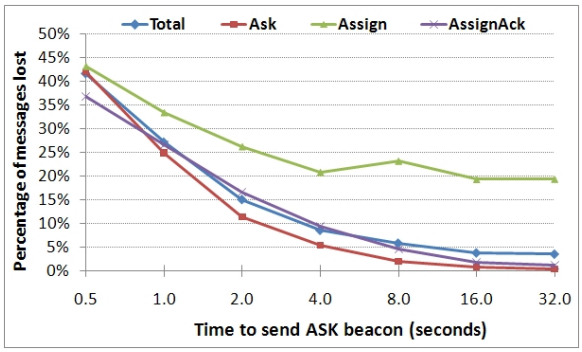
This section presents a parametric study performed to explore the different parameters in the KSP. The list of all parameters used are shown in the Table 5.3. Only one of this parameter was varied while others were kept constant. This allows finding the most suitable value for a particular parameter, to acheive the best performance.

5.5.6.1 Effect of Time to Send ASK (Delay) on KSP

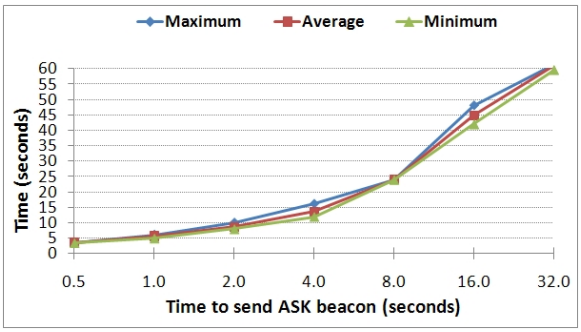
The ASK beacon is sent by nodes after a random time from a span of values. In these experiments this time span is varied. The values are varied from 0.5 to 32 seconds in a binary exponential manner. Note that the smaller the value, the quicker will the KSP finish. The other parameters are constant and appear in Table 5.3. The percentage of messages lost and

Table 5.3: Parameters used in different experiments on Key Setup Phase parameters

Parameter	Standard Value	Variation
Number of experiments	10	5 to 10
Number of Nodes	100	100
Node Density	16 Average	4 to 26 Average
Noise conditions	No noise	No noise
Time span for ASK	4 seconds	0.5 to 32 seconds
Time span for ASSIGN	50 msec	6.25 to 400 msec
Number of attempts to send ASK	1 time	1 to 2 times
Number of attempts to establish neighbours	5 times	0 to 6 times



(a) Percentage of message lost in KSP as a function of ASK delay

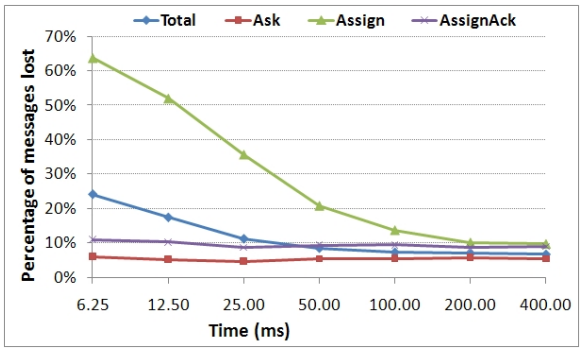


(b) Time taken to complete KSP as a function of ASK delay

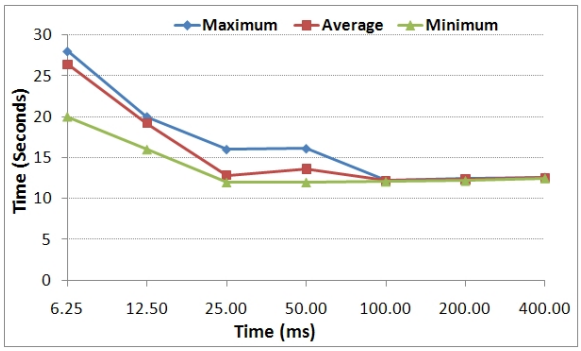
Figure 5.10: The effect of time to send ASK (delay) in the KSP

time taken to complete KSP are shown in Figure 5.10(a) and Figure 5.10(b) respectively. As there is no noise model in these simulations, all the message are lost as a result of collisions. The message loss graph indicates the average total messages lost in KSP after performing 10 experiments. The average ASK, ASSIGN and ASSIGNACK messages lost are also plotted.

It is evident by looking at the graph in Figure 5.10(a) that the collision is reduced when the time span to send ASK is increased. The average total message loss is below 5% when the time is increased to 16 seconds. It was also observed that a similar affect is evident on ASK and ASSIGNACK beacons but not on ASSIGN beacons. The message loss (collisions) in ASSIGN beacon decreases in the same manner up till 4 seconds. If, however, the time is increased further there is no further change and the message loss remains around 20%. This means there is another factor affecting this message loss (collision). That will be discussed in a later section.



(a) Percentage of message lost in KSP as a function of ASK delay



(b) Time taken to complete KSP as a function of ASK delay

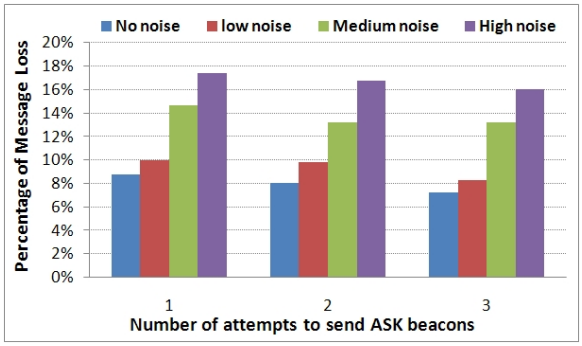
Figure 5.11: The effect of of time to send ASSIGN (delay) on Collision in the KSP

Although the increase in time has decreased collisions considerably, one has to pay the price in the form of more time to complete the KSP and increased vulnerability to attacks. This is evident from Figure 5.10(b) where the setup time increases. Also, there is not much variation in the results between experiments. Note that the standard total time must be less than 3 times the span of an ASK beacon (ASK is sent 3 times by each node). It also depends on collisions, as the nodes try to resend the messages that are lost in the collision. When the time span is increased to 4.0 seconds, the time to complete is almost 3 times the ASK span time. Considering collisions again, in Figure 5.10(a), shows that it is because the collision is reduced to below 8% in these cases, thus supporting the claim.

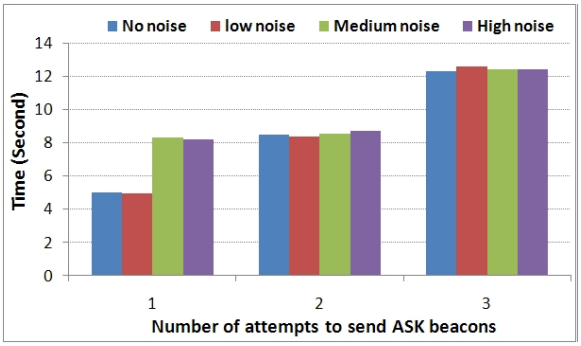
5.5.6.2 Effect of Time to Send ASSIGN (Delay) on KSP

It was evident from Figure 5.10(a) that there is a factor causing the collisions in ASSIGN beacons. In these set of experiments the time span to send ASSIGN was started from 6.25 msec and was doubled up to 400 msec; the other parameters were kept constant. The time span for ASK was fixed to 4 seconds as it had given a good performance in both collisions (<10%) and setup time (12 seconds). Considering the Figure 5.11(a), it is evident that the collision in the ASSIGN beacons decreases considerably with the increase in the time span for the ASSIGN beacon. The collisions on both ASK and ASSSIGNACK beacons remain constant, but the overall collisions decreases because of the decrease in the collisions in ASSIGN beacons. It is also evident that after 200 msec time there is not much decrease in collisions. Therefore it appears that, in high density networks, all the collisions cannot be eliminated.

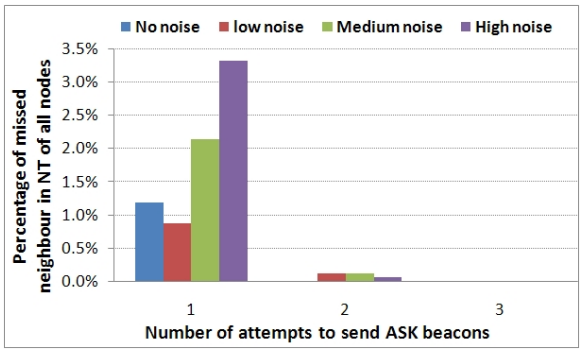
The effect of the ASSIGN time span on completion time for KSP is shown in Figure 5.11(b). It is evident that this time the effect is the opposite to that of the ASK time span. The time to complete the KSP is quite high until the span for the ASSIGN is reduced to 25 msec and then



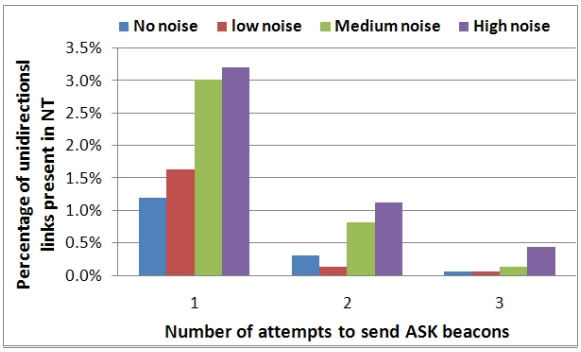
(a) Percentage of message lost in KSP



(b) Time taken to complete KSP



(c) Percentage of missed neighbour in NT of all nodes



(d) Percentage of unidirectionsl links present in NT

Figure 5.12: The effect of sending redundant ASK messages in KSP

it stays constant. This results from a lot of messages being lost because of collision if the time span is low. Thus, the protocol will try to resend more messages to establish the KSP. This will eventually increase the time to complete the KSP. It is already evident that collisions are greater if the time span is less than 25 msec. This indicates that if the ASSIGN is sent before 25 msec, no advantage is achieved because most of the messages are lost because of collision and KSP will also not be completed early.

5.5.6.3 Effect of Redundancy in ASK

The effect of collisions have been reduced considerably by adjusting different parameters. A further series of test was focused on checking the effect of the number of times the ASK beacon is sent. Consider that the ASK beacon is the first beacon a node broadcasts after it is booted. It is never a guaranteed that all the neighbours will receive the ASK beacon. So, in order to provide redundancy, the ASK beacon was sent multiple times. It has a side effect though. The setup time is directly proportional to number of times ASK is sent. So by broadcasting the

ASK twice, the setup time will be doubled. Thus, there must be a justification for sending ASK more than once. The experiments were performed in the presence of different noise samples as described in Table 5.2. Other parameters of Table 5.3 were unaltered.

Considering Figure 5.12(a) it is clear that message loss is higher when the noise is high. There is a slight increase in message loss as the number of attempts to send ASK is reduced from 3 to 1. This time the important thing to check, instead of message loss, was non-existent entries in NT. These are the number of neighbour nodes missing in NT of different nodes (Figure 5.12(c)). It is evident that sending the ASK beacon once has created more problems in the NT of nodes. The error increases if a high RF noise sample is used (3.5%) and quite low in the absence of RF noise (0.5%). In the absence of noise, errors results form collisions alone. The results are far better if the ASK is sent twice. In the presence of noise samples the error rises to a maximum of 0.6%. When the ASK beacon is sent 3 times, there is not a single error entry even in the presence of high noise.

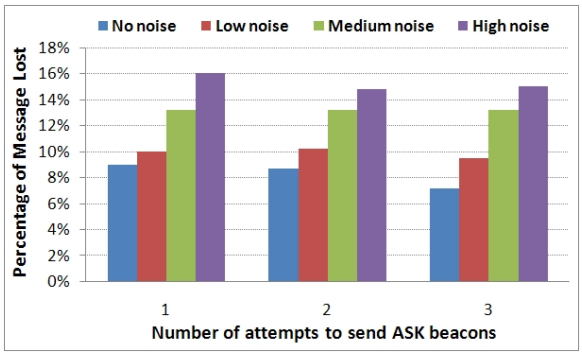
It is necessary to check the number of unidirectional links present in the NT, i.e. neighbours that were not verified due to message loss in the KSP; the results are shown in Figure 5.12(d). Again, the percentage loss is more if the ASK is sent once especially under high RF noise conditions. The lowest error percentage were achieved when the ASK was sent 3 times; although in the presence of high noise, the error is only 0.5%. This is negligible considering the high number of neighbour nodes.

Finally, the time taken to finish the KSP is examined for different cases in Figure 5.12(b). The expected time to finish should be 4 times the number of attempts made to send ASK (ASK time span is 4 seconds). Considering the graph, most of these followed this trend except in the case when ASK is sent once and the noise sample is medium or high. This is because in the presence of noise, the messages are lost more often and the protocol tries at least one time to rebuild the unidirectional links in NT (ASSIGN attempts=1). This will increase the setup time. Note that this has already been verified by the Figure 5.12(d), where ASK is sent once, the errors are greatest under in medium and high noise conditions.

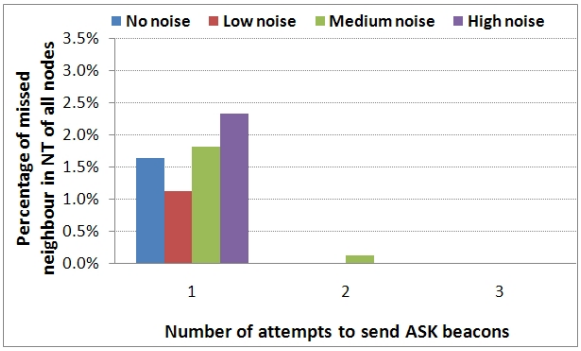
5.5.6.4 Effect of Redundancy in ASK with 2 Assign Attempts

The only difference in this experiment compared to the previous experiments was that 2 attempts were made to establish unidirectional links. The ASK beacon attempts ware again varied between 1 and 3 with other parameters kept constant. However, the 4 different noise samples were also applied. As expected, the results of message lost (Figure 5.13(a)) remained constant.

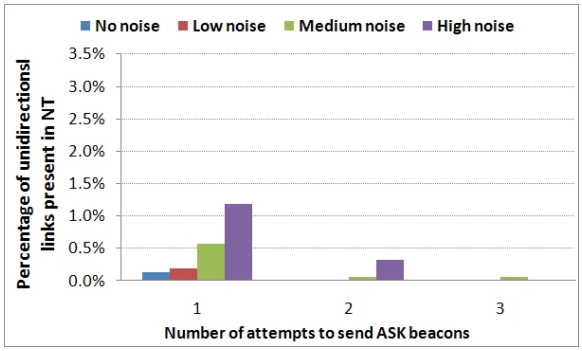
The errors in the NT, i.e. the number of neighbour nodes missing in the NT of different nodes, is much lower than in the previous case, as shown in Figure 5.13(b). The error this time is lower than 2.5% in the presence of high RF noise when sending the ASK only once. There is a little improvement when the medium noise sample is employed. It is worth noting that



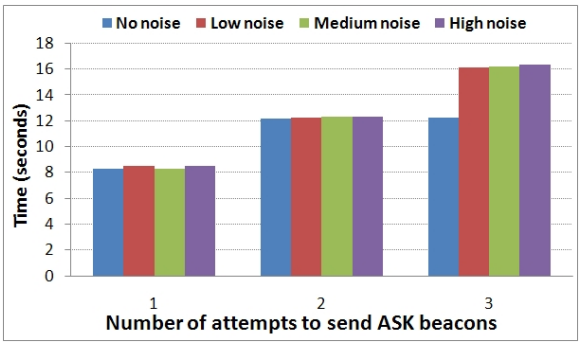
(a) Percentage of message lost in KSP



(b) Percentage of missed neighbour in NT of all nodes



(c) Percentage of unidirectionsl links present in NT



(d) Time taken to complete KSP

Figure 5.13: The effect of sending redundant ASK messages in KSP

the error remains the same for low noise. It increased however when the no noise sample is employed. The reason for this may be that messages collide more under no noise conditions. In the presence of low noise conditions, some of these collided messages are lost due to RF noise, enabling at least a few to reach their destination. Note that data traffic has also been increased by sending redundant ASSIGN beacons. Thus they may have collided with the only ASK beacon sent. The results get better with the increase of the number of the ASK beacons sent. As in the previous case, the NT entries are 100% correct when ASK is sent 3 times.

The number of unidirectional links present in the NT (i.e. neighbours that were not able to be verified due to message loss in the KSP) has also decreased a lot by trying to attempt twice to establish the unidirectional links; the results are shown in Figure 5.13(c). Comparing this with the previous case in Figure 5.12(d), the error has reduced and when 3 ASK beacons are sent the results are ideal even in the presence of high RF noise.

Finally, the time taken to finish the KSP is plotted for the different ASK beacons attempts in Figure 5.13(d). As stated earlier, the expected time to finish should be 3 times the number of

attempts made to send the ASK beacon. As the ASK time span is 4 seconds, the standard time should be 4, 8 and 12 for 3 cases respectively. Looking at the graph, in almost all cases, the protocol took extra time to complete the KSP. The reason being, in case of unidirectional links, the network tries to establish paths with unidirectional nodes twice and so times accumulate. Therefore, the finish time might go up to a maximum of 5 times the ASK time span. When there is no noise and the ASK is sent 3 times, the protocol was able to complete within the expected time (12 seconds). During this period all nodes establish the correct entries in the NT in the absence of RF noise and leads to KSP finishing in time. It is not so in the presence of noise because some of the messages are lost and nodes again need to re-establish the unidirectional links. Note in Figure 5.13(c) that it is clear that when ASK is sent 3 times the unidirectional links are less 0.1% even in high noise.

Finally, the number of attempts to re-establish the incorrect links should not be increased too much because, in the presence of hello flood attacker, it will lead to a lot of message overhead. That is the reason that the experiments were not performed for more than 3 attempts for redundancy.

5.6 Route Setup Phase (RSP)

The main function of the RSP is that the nodes know their own as well as their neighbour's relative hop distance from the BSs in an authenticated way. Thus, a node captured in this phase will not affect wrong/false message propagation. Another task of the RSP is that the virtual links, because of INA and wormhole attack, are avoided.

The general assumptions for RSP are:

- Links may be unidirectional or bidirectional.
- A node can be captured by attackers in this phase.
- The KSP has been completed before this phase.
- The base stations can be one or more.
- The one way hash chain used by the the BSs cannot be decrypted.
- The BSs are in safe locations and cannot be captured.
- Network scalability and density may vary.
- High noise spikes might be present and cause considerable message loss.
- The hidden terminal problem exists at the link layer.
- A square grid topology is used for the simulation throughout.

- For certain cases, the messages are sent with a higher power than normal.

This section is further organized as follows: The available techniques that can be used in the RSP are briefly discussed in Section 5.6.1; the scheme adopted by RAEED for the RSP is explained in Section 5.6.2; the four sub-phases of the RSP NPP, LTP, NSP and LPP are explained in Section 5.6.3, Section 5.6.4, Section 5.6.5 and Section 5.6.6 respectively. The message sequence diagram for the RSP is explained in Section 5.6.7 and a formal model to verify the correct working of RSP is explained in Section 5.6.9. Computer simulation are later performed to confirm the working of RSP without a synchronization process (Section 5.6.10), with a synchronization process (Section 5.6.11) and in the presence of multiple BSs (Section 5.6.12).

5.6.1 Available Techniques for Route Setup

There were many different techniques discussed in the literature review to route data to the sinks or BSs. Some of these are: (i) flood data by broadcasting, (ii) send data to the node that is nearest to the sink by using location, hop count, cost, and low latency, (iii) assign a parent in the setup phase and always forward the data to the parent. and (iv) Send data randomly to a neighbour.

Flooding data is not normally employed by routing protocols as it is the worst case of data routing. Sending the data by randomly selecting a neighbour may cause the data to take longer route and may result in the message loss in some cases. Our experiments on INSENS confirmed that sometimes old messages collided with the messages arriving later and thus both messages are lost. Sending data to a parent is a good technique but will fail if the parent has died or has been compromised. In that case the parent becomes a single point of failure. Although solutions like the Parent Monitor scheme [150], to monitor parents if they are still active do exist, they cannot avoid node compromise and cannot be regarded as secure. The technique most often used by routing protocols is forwarding data to the node nearest to the sink/destination or BS. These schemes are identical to forward data to a parent. The node nearest to the sink uses one of the following methods to determine the next hop node: (i) geographical distance, (ii) hop count distance, (iii) message latency or time taken to receive message and (iv) cost in terms of energy etc.

Geographical distance normally involves GPS or some other technique to measure the actual locations of nodes. As RAEED avoids any additional hardware, this technique has not been adopted. The hop distance is a technique in which a tree is built up starting from the BS (sink) to all nodes, where the BS is on top of the tree. In this case, the BS initiates the route broadcast which is propagated by each node once and ignored the next time. A similar technique uses time as a metric rather than the hop distance. The nodes try to forward the data to the node which provides the least latency. In that case the time taken to receive a message involves the synchronisation of the nodes. The nodes might be synchronised before deployment or by some

authentic manner. The last option uses energy or cost involved in message passing as a metric for data routing. This metric is used to indicate how much cost is required by a node in terms of power etc. Note that false information might also be propagated in such cases.

5.6.2 The Proposed Design

RAEED adopts a combination of the time taken to receive a message and the hop count technique. Both are simple and require no additional hardware. Moreover, the authentication can be provided in this technique; this is discussed in later sections. The fields of a message used in RSP are:

$$Type, SenderID, TargetID, Nonce, PairKey, ClusterKey, Time, HMAC$$

Note that the fields used in the KSP are no longer required so the *Nonce* field can be replaced by the OHC, the new one way hash chain attached in the message. The *PairKey* and the *ClusterKey* fields can be replaced by 8-bit representations of the BS time in seconds and milliseconds. The *Target* field is no longer required in this as the messages are broadcast in the LPP (and NSP). So this field can be reserved for the BS ID (index) if multiple BSs are deployed. Lastly, a new field HMAC is used. This is the hashed message authentication code using a one-way hash chain (OHC) which is stored in all nodes and the BS. So the modified message format in the RSP is:

$$Type, SenderID, SinkID, OHC, Time_{second}, Time_{ms}, Time_{minute}, HMAC$$

The RSP can further be divided into the Neighbour Propagation Phase (NPP), the Node Synchronization Phase (NSP) and the Level Propagation Phase (LPP). The NPP unicasts neighbour information gathered in the KSP to all the neighbouring nodes. The NSP involves synchronising the nodes by flooding the authenticated message from the BS using a OHC. The LPP involves assigning the nodes hop distance levels from the BS in an authenticated manner. The NSP is a precursor to the LPP as the LPP presented in this thesis requires the nodes to be roughly synchronized. The NPP can take place before or after the combination of NSP/LPP but performing it just after the KSP is advantageous. This will be discussed in later sections.

5.6.3 Neighbour Propagation Phase (NPP)

This phase could be started at any time in the RSP, even after finishing the Level propagation, but sending neighbour information earlier is advisable as it will prevent an attacker from sending fake information after node capture. This will also enable the protocol to remove the wormhole attack using the Loud Test Phase (LTP) presented later on in Section 5.6.4 and Section 6.3 before the Level Propagation Phase (LPP). Thus, the preferred time is to do it straight after the KSP. Three methods by which neighbour information is generally exchanged are:

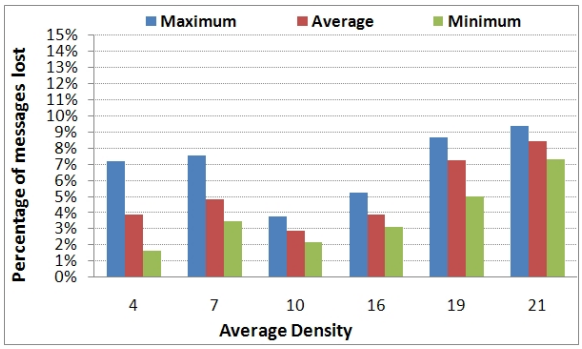


Figure 5.14: The effect of density on percentage of Neighbour message lost in 100 node network

1. First method is to send it randomly in a selected time. This will enable collisions but will be finished very quickly.

2. A second method is to synchronize node clocks and then send the neighbour information according to node ID, i.e. the lower ID should send data first and so on. This will prevent collisions but will increase the time to finish this phase. But this is only possible after the LPP.

3. A third method is that each node divides the time reserved to send neighbours into different slots, depending on the total number of neighbours, and then messages are sent depending on how high the node ID is compared to others. Instead of sending data immediately, a random time might be chosen in this available time slot to avoid further collisions.

Considering the above three options the third one is a combination of the first two options and looks more efficient. It will result in fewer collisions compared to option 1 and will also finish in the same time duration as that of option 1. Hence, this option has been chosen for neighbour propagation. The constant used for time slot is called `MAXNEIGHBORTIME` in the TOSSIM software and the time slot is calculated using the equation:

$$TimeSlot = \frac{MAXNEIGHBORTIME}{TotalNeighbor}$$

Another concern is that the MICAZ motes did not support large size messages and even for 28 neighbour nodes (28x2=56 bytes) multiple messages have to be sent. A possible solution could be to send the difference in node ID in 8 bits but sometimes the node IDs might have values greater than 256. So this is not practicable.

The sender node must also know that it has received all the NEIGHBOR messages from a node. To provide this, a stamp is included in NEIGHBOR message which is simply sum of all neighbour node IDs a node must send. More complexity in stamp is left for future work. Using

this stamp the receiver node can confirm that all node IDs have been received (by adding the received node IDs of the neighbours).

It is expected that some of the NEIGHBOR messages will be lost as a result of collisions. Figure 5.14 shows the effect of density on the percentage of the NEIGHBOR message lost in a 100 node network. Thus some nodes will miss the 2-hop information from some of their neighbours. The data stamp will ensure that the nodes correctly receive all 2-hop neighbour's update. The format of NEIGHBOR message is:

$$N \rightarrow * : (NEIGHBOR, [ID_N, ID_1, Stamp, ID_2, ID_3, ..., ID_{12}]_{K_G})$$

where

$$Stamp = \sum_{i=0}^{n} ID_i$$

Here n is the total number of neighbours and ID_i is the 16 bit ID for ith neighbour stored in the node. The same stamp is attached for multiple NEIGHBOR messages (if total number of neighbours is more than 12). After a suitable time the nodes check if the stamp is correct for a neighbour. If the stamp is incorrect for a particular neighbour a MISSNEIGHBOR is broadcasted:

$$N \rightarrow * : (MISSNEIGHBOR, [ID_N, ID_1, -, ID_2, ID_3, ..., ID_{12}]_{K_{C_N}})$$

This time ID_i contains IDs of neighbour nodes whose stamps are incorrect and thus nodes require them to resend the NEIGHBOR messages. The neighbour nodes check the MISS-NEIGHBOR message and if their ID exists in this message, they rebroadcast the NEIGHBOR message. In this way all the missing 2-hop neighbours can be recovered. Also, the total time reserved for NEIGHBOR message propagation was 1 second so the number of collision has increased. This time is relaxed and later increased to 10 seconds to avoid collisions.

The MISSNEIGHBOR message might be sent anytime even after the LPP is finished, i.e. levels have been assigned. The missed neighbour nodes can then be checked by looking at the neighbour stamp, sent by each neighbour, which was temporarily stored in the NT (Rank field). If the stamp is incorrect, the node builds a MISSNEIGHBOR message by inserting that neighbour's node ID in the message and broadcasts it. Note that these missed neighbours will be omitted in the loud test if the missed neighbours are broadcasted later after LPP. One solution is to broadcast these missed neighbour messages straightaway after NPP, before the BS initiates SPP. Even if the node has been compromised, the attacker cannot later send fake neighbour IDs because the stamp and a few neighbour messages have already been received by legitimate nodes.

5.6.4 Loud Test Phase (LTP)

The LTP scheme assumes that nodes can transmit a powerful message. This will enable a node to transmit the message to a node 2-hop away, which, in normal transmission, is not in radio

range of the node. This assumption is realistic as the node's transmission can be adjusted at run time without adding any extra hardware. The aim here is to use this powerful signal sparingly (at the most once) only to detect the virtual links between nodes.

The main aim of this phase is to remove virtual links created because of wormholes and INA. These issues are explained in detail in Section 6.3. In this section only the design of LTP is presented. Two hand shake messages LOUD and LOUDREPLY are exchanged between 2-hop nodes similar to what has been done in the KSP (ASK and ASSIGN) between 1-hop nodes. This scheme is only possible if NPP has successfully finished, which enables all individual nodes to be aware of their possible 2-hop neighbours. The design is flexible and the LOUD messages can be broadcast or unicast. The LOUDREPLY is always a unicast message and is addressed to a particular node. If the global key (encryption) is assumed to be unknown to intruders until the end of this phase (no node can be compromised before LTP finishes), broadcast LOUD can be used. This will have a low message overhead as a single LOUD message is sent by nodes to all 2-hop nodes instead of individual LOUD message to all 2-hop neighbours.

If the broadcast technique is to be used, the message format of LOUD is the same as used in the ASK beacon. The only difference being that it is sent with more power:

$$N \twoheadrightarrow * : (LOUD, [ID_N, -, nonce, -, -]_{K_G})$$

A receiving 2-hop node, R, updates its NT and responses with a powerful LOUDREPLY beacon which contains the nonce it received from the 2-hop sender node. This message is also encrypted with the global key, K_G. Note that the message must be powerful enough to reach 2-hop neighbour nodes (indicated by $\twoheadrightarrow$ instead of $\rightarrow$). If required, the 2-hop neighbour nodes can also generate and send a new pair key $K_{R,N}$ (as well as new cluster key for 2-hop nodes K_C) which can be later used for communication between these two nodes.

$$R \twoheadrightarrow N : (LOUDREPLY, [ID_R, ID_N, nonce, K_{R,N}, K_C]_{K_G})$$

Note that the above 2 equations are modified by replacing the global key K_G by the corresponding cluster key if the LOUD messages are unicasted instead of broadcast.

5.6.5 Level Propagation Phase (LPP)

An innovative technique, using authentic time stamps, is presented to assign levels to nodes instead of the usual incremental hop count message passing. The BS initiates this phase by broadcasting a LEVEL beacon. It contains the time at which the beacon is initiated, $Time_B$, and next element of the hash chain OHC_1 encrypted using a one-way hash chain (OHC). Here $Time_B$ comprises 3 bytes reserved for time in minute, seconds and milliseconds. Due to the one way property of the OHC, receiving nodes can confirm authenticity of the LEVEL beacon that

it has been generated by a legitimate BS. The complete message is then encrypted again using the cluster key, K_{C_B}, of the BS:

$$B \rightarrow * : (LEVEL, ID_B, ID_B, OHC_1, Time_B, HMAC^{LEVEL}_{OHC_0}]_{K_{C_B}})$$

where

$$HMAC^{LEVEL}_{OHC_0} = ID_B, OHC_1, Time_B$$

and

$$Time_B = Time_{second}, Time_{msec}, Time_{minute}$$

The receiver node N, upon receiving this message, first confirms if the sender has a verified entry in the NT and thus can decrypt the message with the sender's cluster key stored in the KSP. It can then authenticate the message, the time received using one way hash function and its cached OHC number OHC_0. If OHC is valid, the cached OHC number OHC_0 is updated to OHC_1 which will be used by the BS for future authentication. This process is repeated and OHC_1 is updated to OHC_2, OHC_3 etc in the next generated authentic messages by the BS. Because of the one way hash function, a node can authenticate all lower hash values. The node decrypts the time stamp attached by the BS, $Time_B$, and assign itself the level by calculating the time difference between the current time at the node and the BS stamped time. As the OHC has been used, only authentic BS messages will be accepted by the nodes. Thus the nodes will calculate the legitimate level. This is the reason that a timing mechanism is used to assign the node levels. Moreover, the nodes need not be synchronized perfectly as one can use the difference of time (e.g. 100ms has been employed in this thesis) to assign a particular level. This value is saved in each node in a constant called `TIMELEVEL`. Each node N calculates its level by using following equation:

$$Level_N = \frac{Time_B - Time_N}{TIMELEVEL} + 1$$

Each node rebroadcasts the LEVEL message after a random time between 0 and `TIMELEVEL` (100 ms). It is expected that the error between the node clocks is less than `TIMELEVEL`. The whole message is encrypted with node's key K_{C_N} :

$$N \rightarrow * : (LEVEL, ID_N, ID_B, OHC_1, Time_B, HMAC^{LEVEL}_{OHC_0}]_{K_{C_N}})$$

The sender is assigned the level depending on the time stamp. A node only broadcasts the LEVEL beacon once when LEVEL beacon is received the first time from the neighbour node. Later LEVEL beacons merely update the NT giving each node the *level* depending on its node time stamp. This eventually becomes the hop count from the BS as each node broadcasts data after 100ms at most. Nodes then can check the time by looking at the authentic time of the BS (OHC). Thus in spite of node capture before this phase, the attacker cannot propagate a false

level by assigning any value to the hop count. If however a wormhole or INA was present just before deployment, the node can still get incorrect levels. So these two attacks must be avoided before this phase. The method to avoid these attacks is discussed later in Section 6.3. A level is only a guide to each node informing how far this and its neighbours are from a particular BS. Also the nodes do not always send data to the parent node from which it first receives the LEVEL beacon. Hence a parent node's compromise will not fail RAEED.

The receiving node will first find the entry in the NT to get the cluster key to decrypt this message. It then verifies the hash value and the time stamp of BS using the OHC as explained earlier. Then the receiver node reads the time stamp sent by node N to determine how long it takes the message to reach that node. If the sender node broadcasts a fake time it will be detected straightaway and that message will be discarded.

Note that because of the random time, the time to send will decrease as the LEVEL beacon moves forward because the random delay time may be of any value from 0 to `TIMELEVEL`. In this way, the nodes may get incorrect levels. To avoid this happening, each node checks to which time span of `TIMELEVEL` (e.g. 100ms) the LEVEL beacon belongs. It then adds the remaining time in the time span to the delay to broadcast the LEVEL beacon. The time span can be determined by using the node level that has been authentically assigned:

$$TimeSpan_N = Time_B + (Level_N * TIMELEVEL)$$

The error is the difference between this time span and the current time at the node when the Level beacon was received. This error is added to the random time to rebroadcast the LEVEL beacon:

$$SendTime_N = (TimeSpan_N - CurrentTime_N) + (RandomTime + TIMELEVEL)$$

The RSP is explained using a simple example in Section 5.6.8.

5.6.6 Node Synchronization Phase (NSP)

It was realized that lot of errors in level resulted because node's clocks were not synchronized. It had been stated in the assumptions that no special hardware will be used in RAEED. Thus, synchronization of clocks was the issue the current research wanted to avoid. However during deployment the nodes may have had different start times in which case there will be lot of errors in level propagation. Also the researchers have shown that nodes might have a maximum worst drift of 40 ppm. This means 40 us in one second and therefore the issue is to synchronize node clocks somehow before level propagation so that even the worst drift would have no effect on them. To solve this problem, each BS, floods the network with its time stamp before initiating the LEVEL beacon. Each node will then update its time according to that time stamp. This would be authentic as it will utilize one-way hash chain as was in the case of LEVEL beacons.

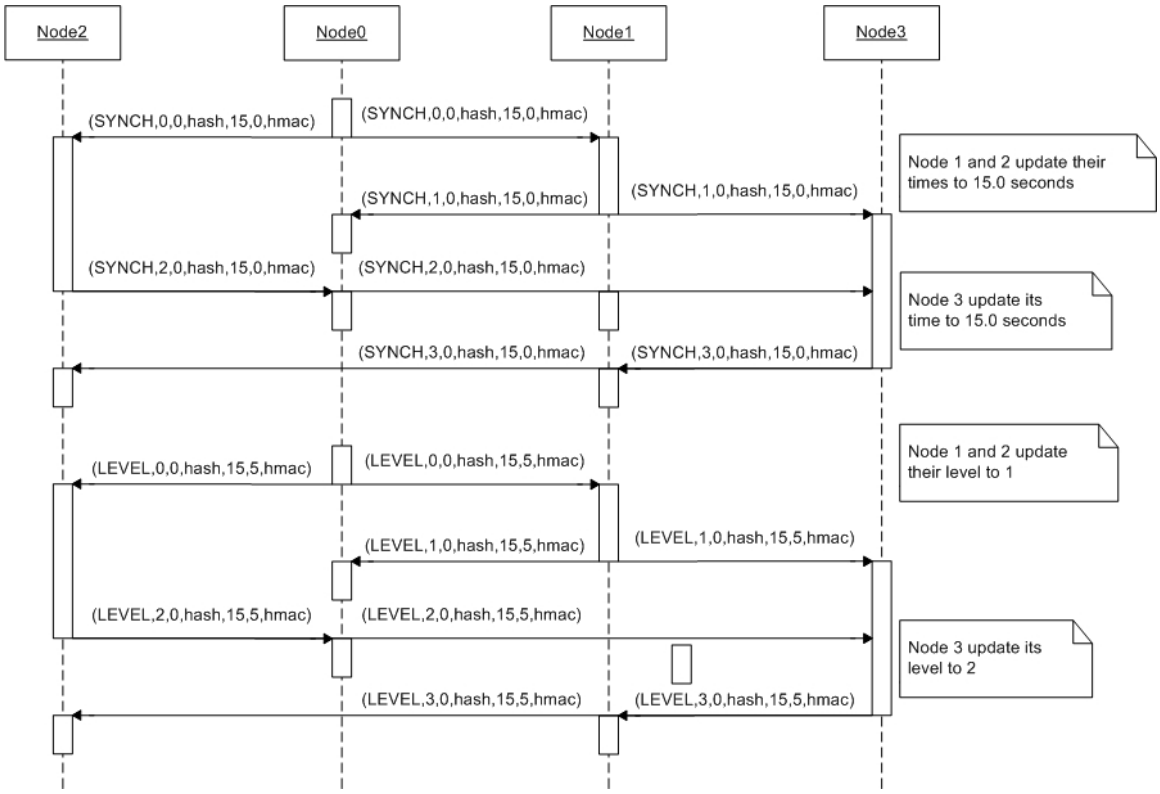


Figure 5.15: Message Sequence Diagram for RSP

The messages are treated in the same manner as in LEVEL beacons i.e. accepting the first message and ignoring the rest. The format of the message will be:

$$B \rightarrow * : (SYNCHRONOUS, ID_B, -, Time_B, HMAC^{TIME}_{OHC_0}]_{K_{C_B}})$$

where

$$HMAC^{SYNCHRONOUS}_{OHC_0} = ID_B, Time_B$$

Nodes simply forward this message without delay. Nodes can confirm authenticity by using the OHC and thus update their time if required be. This synchronization is required only before the BSs propagate the LEVEL beacons. A compromised node delaying the message will gain no advantage as the delayed messages are rejected. The only possible attack is hello flood or wormhole, the attackers propagating this message earlier than the normal propagation time. The hello flood attack has already been removed from the protocol in the KSP and the solution for wormhole will be presented later in Section 6.3.

5.6.7 Message Sequence Diagram

Figure 5.15 shows the message sequence diagram of a 4 node network to demonstrate the SPP and LPP in the RSP. Node0 is the BS that has 2 neighbours Node1 and Node2, which thus must have a level of 1 because they are one hop away from BS. There is another Node3 that is

within the range of Node1 and Node2 but not that of Node0. This node is thus two hops away from BS and therefore must have a level of 2. The message format used for this beacon is (Type, SenderID, TargetID, BaseStationID, NextHash, $Time_{Second}$, $Time_{Msec}$, $Time_{Min}$, HMAC).

The BS initiates the SYNCHRONOUS message with time stamp of 15.0 seconds. Node1 and Node2, upon receiving this message, synchronize their node time according to the BS time, i.e. 15.0 seconds. Both nodes then rebroadcast the message. In this case Node1 has transmitted before Node2. This enables Node3 to receive message from Node1 before Node2 and would have synchronised its clock. Node2 also receives this new SYNCHRONOUS message but ignores it, as the node has already been synchronized. Similarly, later on when nodes Node1 and Node3 receive SYNCHRONOUS message from Node2 they would also ignore it. Finally, Node3 rebroadcasts the SYNCHRONOUS message which will be ignored by both the nodes Node1 and Node2.

The BS, after 500 ms, initiates the LEVEL message with new time stamp of 15.5 seconds. Node1 and Node2, upon receiving this message, check their own clocks and calculate that the message has been sent within 1 second. The clocks of nodes Node1 and Node2 have already been synchronized. Therefore, both nodes will receive the message after almost 15.5s. The difference in time is thus approximately 0. As the difference in time is less than 1 second, compared to a BS's stamped time, both nodes will update their level to 1 and the level of the sender node (BS) to 0 in the NT. Both nodes then wait for 1 second and then both will try to broadcast the message between 1 and 2 seconds. Suppose that Node3 receives the message earlier from Node1, it checks the time stamp in the message (15.5) and compares it with its own time clock. On comparison the Node3 will realize that the message has been sent within a range of 1-2 seconds after the time stamp. The Node3 thus will assign the Node1 a level of 1 in the NT and a level of 2 (one higher than the sender) to itself. On receipt of this message, the Node2 also updates the level of Node1 to 1 in the NT. Similarly, the level of Node2 is updated in the NT of Node1 and Node3, when Node2 sends LEVEL beacon. Finally the NT of Node1 and Node2 are updated when Node3 tries to broadcast the LEVEL in the time between 2 and 3 seconds after the time stamp.

5.6.8 An Example to Explain Level Propagation Phase

The RSP is presented using a simple network shown in Figure 5.16. Suppose the base station B initiates the LEVEL beacon at exactly 15 seconds. The time stamp will then be 15.00. Taking the constant `TIMELEVEL` value of 100 msec, the different time spans for each level and the node IDs that get that level are shown in Table 5.4.

Both nodes 1 and 2 will receive the message at 15+e seconds, where e is the synchronization error in the node times. This error should be small compared to the `TIMELEVEL`. Both nodes will assign themselves the level 1 and check their current time difference from stamp time. As both these times will almost be the same, the nodes add 100-e msec error delay. The reason

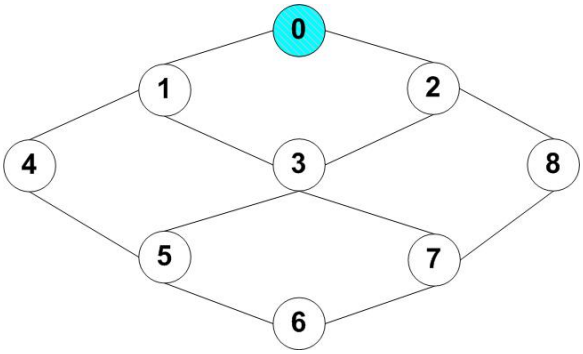


Figure 5.16: A 9 node network to explain level propagation

Table 5.4: Assigned levels and time spans when `TIMELEVEL` is 100 msec

Level	**Time span (msec)**	**Node IDs**
0	15.00	0
1	15.000 - 15.099	1 and 2
2	15.100 - 15.199	3, 4, and 8
3	15.200 - 15.299	5 and 7
4	15.300 - 15.399	6

being that both the nodes must send level message in time span of level 2, i.e. 15.100 to 15.199. Suppose the random time values are 73 msec and 21 msec respectively for node 1 and 2, then node 1 will transmit at 15.173 and node 2 will transmit at 15.121. The receiving nodes are thus assigned level 3 as they receive this LEVEL message between the time span 15.100 to 15.199. Note that the random time is introduced to avoid collisions of the LEVEL messages generated by the same level (hop distance) nodes. The nodes that receive message at 15.121 will add 0.79 msec (15.200-15.121) to the random time before rebroadcasting the LEVEL beacon. This will enable the time to move in the next span i.e. 15.200 to 15.299. This process is repeated throughout and thus all the nodes will get the correct level. For 50 msec value of `TIMELEVEL`, the table is modified to indicate different time spans for each level, The node IDs that get that level are shown in Table 5.5.

Table 5.5: Levels and time spans when `TIMELEVEL` is 50 msec

Level	**Time span (msec)**	**Node IDs**
0	15.00	0
1	15.000 - 15.049	1 and 2
2	15.050 - 15.099	3, 4, and 8
3	15.100 - 15.149	5 and 7
4	15.150 - 15.199	6

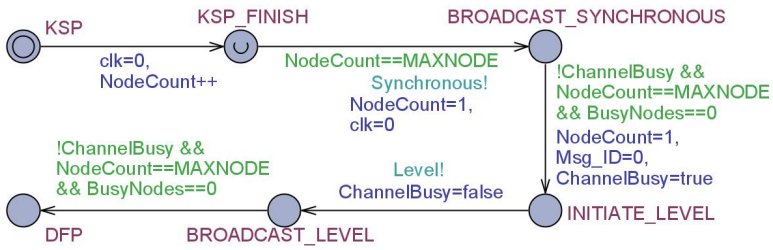


Figure 5.17: Sink model to verify the SPP and LPP of RAEED

5.6.9 Formal Verification of SPP and LPP

The hypothesis for this model is that:

"The nodes can be synchronized using the time stamps and 1-hop neighbours will receive the LEVEL beacon earlier than 2-hop neighbours".

Apart from the assumptions stated in Section 4.3.3, it is assumed that the time tick is in milliseconds and errors less than 1 ms are ignored. Also instead of all 5 node networks, a single 9 nodes grid topology is tested.

5.6.9.1 Model

The model is composed of a node model and a sink model shown in Figure 5.17 and Figure 5.18 respectively. In this formal model the sink model is acting as both sink and event generator.

The ***sink model*** starts in the KSP location. It then moves to the KSP_FINISH location and waits there until all node models end up at the KSP. Before moving to next location the model resets/restarts its clock (`clk`). The sink model waits for some time, using the guard (`NodeCount==MAXNODE`) before initiating the SYNCHRONOUS message. This is done so that all the node models move to the LISTEN location from the KSP location. The `NodeCount` variable is used to track the number of nodes that have finished the KSP. The sink model then broadcasts the time synchronization message (BROADCAST_SYNCHRONOUS). The sink model stays in this location until all nodes are synchronised (`NodeCount==MAXNODE`), the channel becomes free and all nodes go back to their LISTEN locations (`BusyNodes==0`). The sink model then builds LEVEL beacon (INITIATE_LEVEL) and broadcasts it (BROADCAST_LEVEL). Finally the sink waits for the same guard and then moves to the DFP location when all nodes receive LEVEL beacon. This location indicates the end of the RSP.

The ***node model*** starts non-deterministically by moving out of KSP location and thus enter-

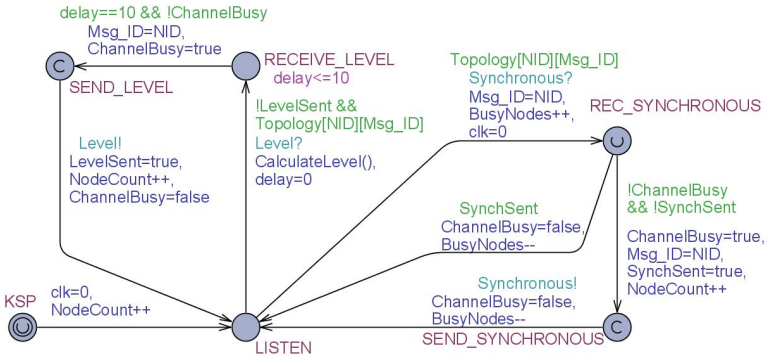


Figure 5.18: Node model to the SPP and LPP of RAEED

ing the RSP. The model then remains in the LISTEN location. If a SYNCHRONOUS message is received (REC_SYNCHRONOUS) from the sink or any other node, the node model will update its clock and rebroadcasts the SYNCHRONOUS message, (SEND_SYNCHRONOUS) if it has not been sent earlier (`SynchSent` flag is false). The channel remains busy throughout this process disabling any other communication during this period. Note that clock is initialized to 0 whenever this SYNCHRONOUS beacon is received.Typically this error may be in microseconds. This being negligible, and can be ignored. When a node or sink broadcasts a LEVEL beacon, the node moves to the RECEIVE_LEVEL location if it has not sent/received the LEVEL beacon before (`LevelSent` is false). The level is calculated using the function `CalculateLevel()` in this transition. It then waits for exactly 10 clock ticks (milliseconds) and if the channel is free, it tries to broadcast the LEVEL beacon immediately. The delay clock in the node is reset when the node receives its level. This clock is used to indicate differences between clocks in different nodes when verifying properties.

5.6.9.2 Verification

Following claims have been verified using properties:

1. The RSP will finish
2. The sink will be synchronized with all nodes
3. The sink will initiate the LEVEL only after the node clocks have been synchronized
4. The lower hop nodes will always receive the LEVEL beacon earlier
5. The clock difference between two hop nodes is always less than the maximum time reserved for each hop

6. The clock difference between nodes 3-hop away from the BS is always less than the maximum time reserved for them

7. The clock difference between nodes 4-hop away from the BS is always less than the maximum time reserved for them

8. All nodes get the desired correct level

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.2.4. All the above claims, including the liveness (Claims 4-7) and safety check(Claims 8), were proved true. This confirms the hypothesis that the nodes can be synchronized using the time stamps and 1-hop neighbours will receive the LEVEL beacon earlier than 2-hop neighbours

5.6.10 Simulation Results without Synchronization Propagation

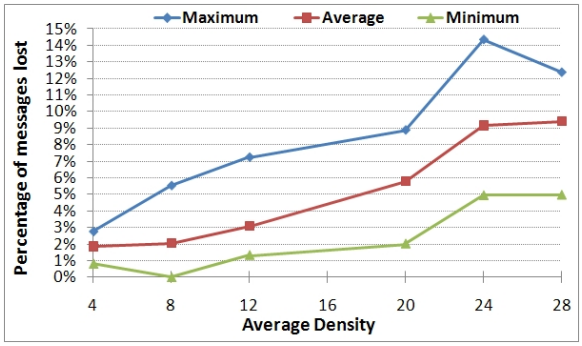
The initial simulation experiments were performed when SYNCHRONOUS message was not induced in the protocol and the nodes were assumed to be turned on at almost the same time. The parameters used for these experiments are shown in Table 5.6, with the parameter name, its fixed value and the full range between which the parameter value is varied.

Table 5.6: Parameters used to perform simulations without synchronization propagation in RSP

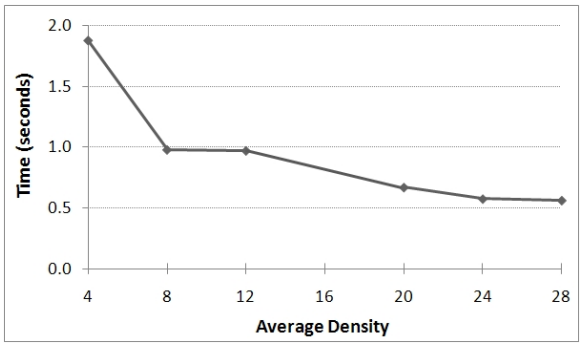
Parameter	**Standard Value**	**Variation**
Number of experiments	5	5 to 10
Maximum number of nodes	100	100 and 1000
Maximum number of BSs	1	-
Node Density	20	4 to 28
Noise conditions	no noise	-
Time span for LEVEL	100 msec	100 to 1000 msec
Percentage of time span used to send	75%	10% to 100%

5.6.10.1 Effect of Density

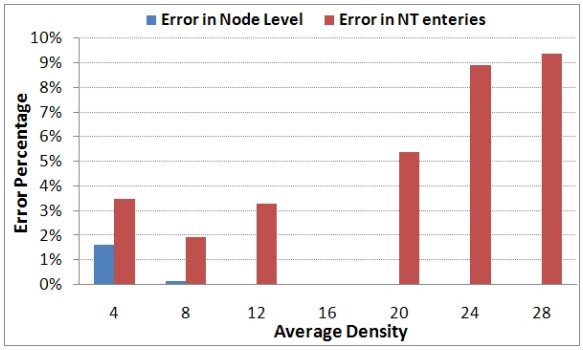
The previous experiments (Section5.5.3) on the KSP have shown that the density has a substantial effect on the results. So the first set of experiments involve varying the density for a 100 node grid network. It is expected that the LEVEL messages lost due to the collisions will increase as the density is increased. The results are shown in Figure 5.19(a), where average, maximum and minimum values of results are plotted. It confirms that the percentage of message lost increases with density.



(a) Error percentage in RSP



(b) Setup Time in RSP



(c) Error percentage in node levels and NT entries

Figure 5.19: The effect of density on level propagation in RSP

The time to complete the RSP reduces as the density is lowered as is shown in Figure 5.19(b). The reason being that the node's hop distance from the BS reduces as the density is increased. The time to complete this phase is directly proportional to the hop distance. So in case of high density, this phase will finish early because the nodes have less hop distance (level) from the BS.

The effect on the percentage of messages loss is symmetric to the error percentage in these experiments. The reason being that only the LEVEL beacons are considered in the experiments so the message loss will be similar to the errors in the NT which is shown in Figure 5.19(c). This is so because the messages which are lost will cause an error or missing entries in the NT. It was observed that in most of the cases the incorrect level assignment was 0 (no level), confirming that all the error in the NT is due to messages lost. Looking at the graph, about 1% percent of the nodes get incorrect level in case the density is lowest(4). Otherwise the error is almost 0%. The reason being that some correct level beacons have collided due to hidden terminal problem when the density was 4. This happened when the two low level nodes simultaneously sent the level. Thus the receiver node received the levels from neither. It then had to rely on

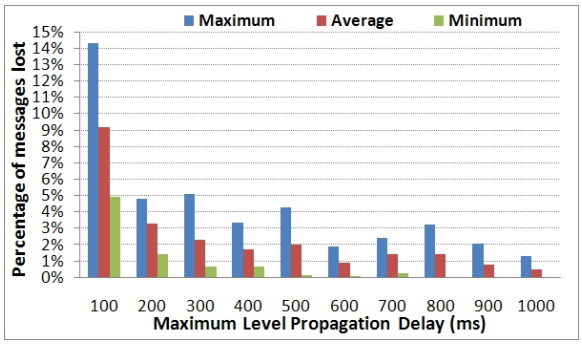
the remaining 2 neighbours to get its level. As the other 2 nodes in that case will have a level higher or equal to the current node so the node may eventually get a level error of 2 (a level 2 more than what it actually should be). Note that the error in the node level is reduced to 0 in higher densities because a node eventually has more chances of getting a correct level. The reason being that in spite of collision, more neighbours are available to it now.

The level error percentage in the NT entries increases with the density. It was expected because high density enabled more level messages lost due to the collisions. Thus level for some of the neighbour nodes cannot be updated correctly. Also note that in case of high error (higher densities), the time to complete RSP is as low as half second. This means that all the messages are exchanged within 500 msec. This automatically increases the probability of collisions. These collisions can be reduced by increasing the value of `TIMELEVEL` which will enable more time to complete this phase but will also reduce collisions. Almost the same error values in case of density 24 and 28 resulted because in one of the experiments of 24-node density the number of collisions are very high (Figure 5.19(a)). This increases the error in the entries and thus increases the overall average error as well. Finally the results for the lowest density i.e. 4 are different from the normal pattern. This is because of hidden terminal problem which has already been explained.

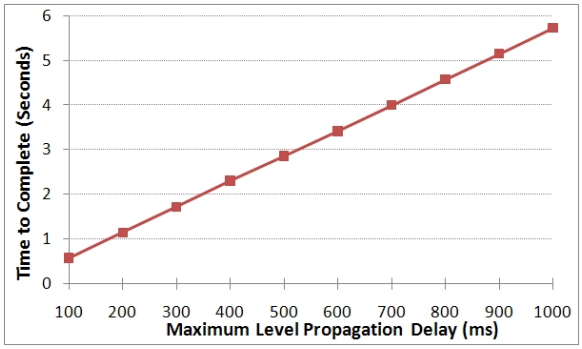
5.6.10.2 Effect of Level Propagation Delay

These set of experiments check the effect of *time span for level*. This is the maximum time (defined as a constant `TIMELEVEL` in TOSSIM software) within which the node tries to rebroadcast the LEVEL beacon. The percentage of message loss is shown in Figure 5.20(a) and as was discussed in last section, it is symmetric to the error in NT shown in 5.20(c).

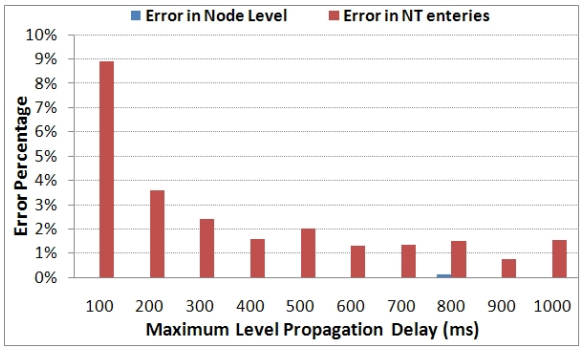
Looking at the Figure 5.20(b), the time to complete level propagation phase (LPP) increases with time. This was obvious as the time span to broadcast level has been increased and obviously it will take more time to complete. It was expected that as the time span is increased, the chances of message collisions will decrease because nodes can now choose from a longer time span. Thus the chances of nodes getting the same random time are less. Looking at the Figure 5.20(c), the error percentage decreased with increase of time span initially but after the time span of more than 400 msec the error percentage stayed unchanged (minor change). The reason being that some spikes (maximum error values) in the graph, which are still quite high, cause the average error to rise. This is so because no random system can guaranty different values and no matter how large the span is, there are still chances of some values remaining the same. However the reduction of minimum value indicates that as the time span is increased, the chances are that less number of LEVEL messages are sent at the same time and thus less collisions. Note that the minimum value becomes 0 from 800 msec time span and the maximum and average values reduced considerably for the last 2 values (900 msec and 1000 msec). It was also noted that when the time span is 1000 msec the values of the message lost and the



(a) Percentage of Level messages lost



(b) Setup Time in RSP



(c) Error percentage in node levels and NT entries

Figure 5.20: The effect of Level propagation delay on RSP

error in the NT were different. The message loss was always 0.8% (negligible) confirming the error was due to incorrect calculations for some neighbour nodes. So future value of 1 second is recommended for the level propagation time span (`TIMELEVEL`). But note that the time to finish the LPP will always be high and can be expressed using following equation:

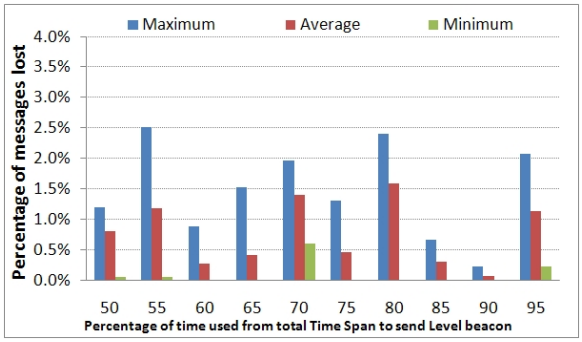
$$TimeSpan_{Max} = HopDistance \times \texttt{TIMELEVEL} \tag{5.1}$$

The minimum value can be calculated using:

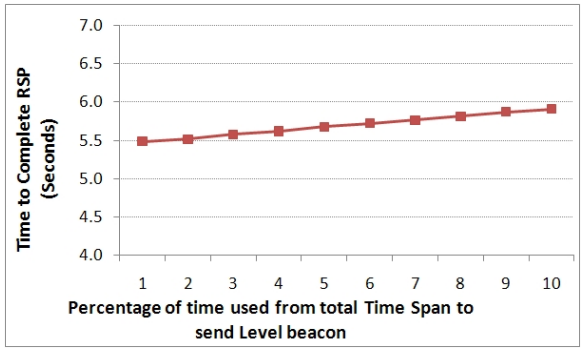
$$TimeSpan_{Min} = (HopDistance - 1) \times \texttt{TIMELEVEL} \tag{5.2}$$

5.6.10.3 Effect of Utilization Time

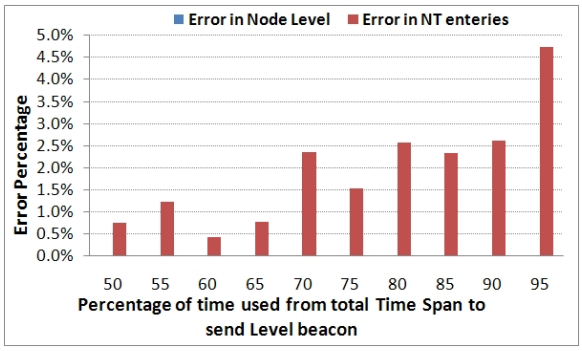
These set of experiments check the effect of percentage of the time span utilized. Note that the complete level propagation delay is set as constant (1000 msec) and instead of using full 1 second to send the LEVEL message, a time varying between 500 msec to 950 msec is used to broadcast the LEVEL. This is effectively 50% to 95% of the actual time span. Figure 5.21(a) shows the percentage of LEVEL beacons lost due to collisions, Figure 5.21(b) shows the time



(a) Error percentage in RSP



(b) Setup Time in RSP



(c) Error percentage in node levels and NT entries

Figure 5.21: The effect of utilization time on RSP

taken to complete level propagation and Figure 5.21(c) shows the average error percentage of level entries in NT in the complete network. As expected the time to complete will be increased slightly as the percentage is increased. It is interesting to see that the values of message lost and error remain the same till the utilization is 65% and then the error percentage is increased. This is so because the nodes are not synchronized and the nodes thus get incorrect level. Note also that the collisions are lowered at higher value except at 95% where the messages do collide. This is because of unsynchronized nodes i.e. two different LEVEL beacons collide with one another. These results prove that in order to get improved functioning the node synchronization is a must. Finally the node error is reduced to almost 2% for all other cases.

5.6.11 Simulation Results after Synchronization Propagation

It had been observed in the previous experiments that some kind of synchronization is needed. Otherwise lot of messages were lost because of collisions and nodes attained incorrect levels. These experiments thus involve the *SYNCHRONOUS* message which were explained earlier in Section 5.6.6. The parameters used for these experiments are almost the same as those in

previous experiments, in Section 5.6.10 and were described in Table 5.6. The only addition being the time span reserved for SYNCHRONOUS message i.e. the maximum time between which the SYNCHRONOUS message is sent randomly. All the parameters for these experiments are listed in Table 5.7.

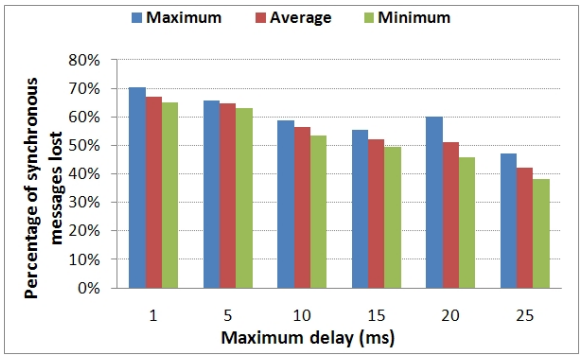
Table 5.7: Parameters used in experiments performed using the Synchronization Propagation in RSP

Parameter	**Standard Value**	**Variation**
Number of experiments	10	5 to 15
Number of Nodes	100	100 and 1000
Number of BS	1	-
Node Density	24	-
Noise conditions	no noise	-
Time span for LEVEL	1 second	500 msec and 1 second
Time span for SYNCHRONIZE	10 msec	1 to 25 msec
Percentage of time span used to send	95	-

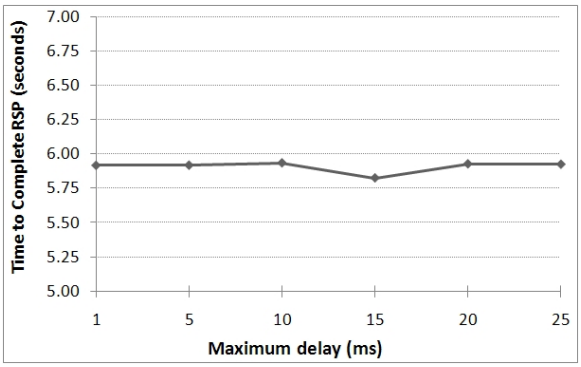
5.6.11.1 Effect of Synchronous Propagation Delay

Figure 5.22(a) shows the percentage of SYNCHRONOUS beacons lost due to collisions; Figure 5.22(b) shows time taken to complete the RSP and Figure 5.22(c) shows the average error percentage of level entries in NT in the whole network. The time to complete remains almost the same with a little increase as the levels are sent after one second and the synchronous delay time span is increased from 1 msec to 25 msec. This is negligible when compared to 1 second. The collision percentage is decreased as the time delay span for the synchronization propagation is increased. This was expected as nodes now have more time span to choose and thus probability of 2 or more nodes sending it simultaneously decreases.

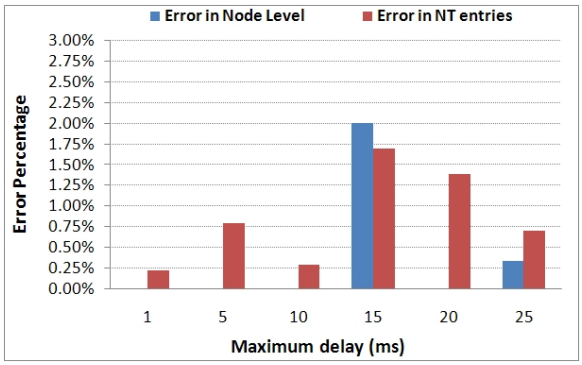
Finally, the result of the NT error entries, which is mostly due to collisions in LEVEL messages, has varying results for different values. The results are better (lesser error) for smaller values of synchronous time span (1, 5, and 10). This is so because lower the synchronous time span, lesser is the difference of clocks of nodes; meaning timers of nodes are enabling nodes to an improved synchronization. At 1 msec value the clocks have a maximum error of 1 msec plus the delay due to MAC layer. But this means sending the SYNCHRONOUS message immediately. As all the values gave similar results so a value of 10 msec is chosen for future experiments. Note that a lower value must be preferred to reduce error in the node times. Due to high collisions the first 2 values (1,5) are rejected and 10 is selected as the optimal value.



(a) Percentage of SYNCHRONOUS beacons lost in RSP



(b) Setup Time in RSP



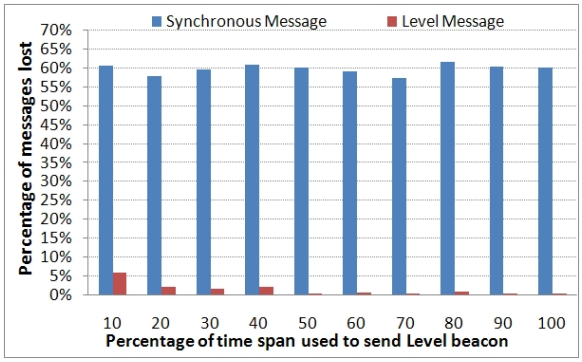
(c) Error percentage in node levels and NT entries

Figure 5.22: The effect of delay in sending the SYNCHRONOUS message on RSP

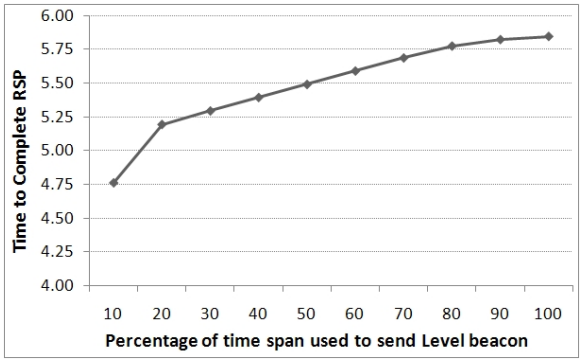
5.6.11.2 Effect of Utilization Time

These set of experiments repeat the effect of percentage of time span utilized for LEVEL messages in the presence of synchronization. This is similar to what has been done in Section 5.6.10.3. Note that, as expected, the nodes are not perfectly synchronized so the clocks of nodes will drift. If the time span is selected randomly, the values might contain the border maximum values. This may induce an error in levels especially if neighbour node's clock is moving in advance. So the border maximum values must be avoided. But selecting a very low value will limit the chosen range. Hence a value around 70% - 80% is expected to give better results. But to reconfirm, different values are plotted in these experiments. To view the effect in a clearer way, the experiments were carried out using two time constant spans. First with the normal span of 1 second was used and then the time span for level was reduced to half i.e. 500 msec. Other parameters remained the same as described in Table 5.7. The results are plotted for 1 second time span.

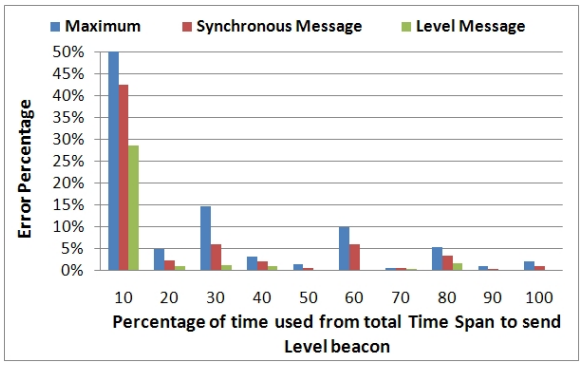
Looking at Figure 5.23(b), which shows time taken to complete the level propagation, it is



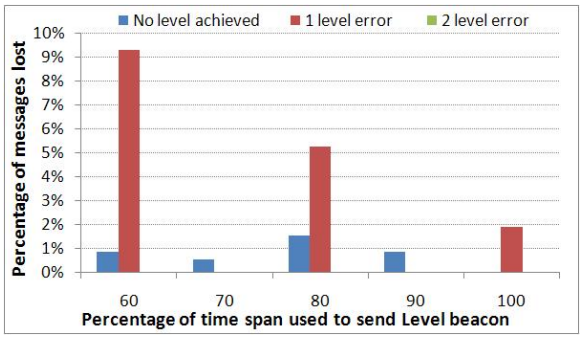
(a) Percentage of SYNCHRONOUS beacons lost in RSP



(b) Setup Time in RSP



(c) Error percentage in node levels and NT entries



(d) Error percentage NT entries: Different catogeries

Figure 5.23: The effect of different time spans used to send LEVEL beacons in RSP

evident that the completion time increased slightly as the percentage of time span is increased. The difference is not much as only the last level nodes (furthest from the BS) can get level earlier than the desired time. The maximum level is 5 so the nodes will finish this phase at $4 + (Percentage \times 1second)$ time. The graph in Figure 5.23(b) confirms that.

Figure 5.23(a) shows the percentage of lost SYNCHRONOUS beacons which almost remained the same for all values indicating that these experiments had no effect on them. Again, similar to the last experiment, the percentage is quite high because of flooding in high density in such a small time span. The percentage of LEVEL beacons is quite low and also it decreases as the time span has increased.

Looking at Figure 5.23(c), it is evident that error in both levels and neighbours level in the NT is quite high for 10% utilization of the time span, i.e. during the rest of the 90% time (900 msec), when the nodes did not send LEVEL beacon. There are many reasons for that. With high density lot of messages will be exchanged in 100 msec resulting in lot of collisions. In 100

msec the nodes may not be synchronized completely (the error of the node time can be up to 100 msec). Thus the nodes may broadcast levels very early, say within 1 msec, and that will induce a level error of one. As the time spans are divided into spans of 1 second, it will also enable collisions of 2 different level nodes in the presence of such a high density. But main reason in the error is the assumption of incorrect level.

For the span percentage greater than 10% the error is quite low. There are some higher values at 30%, 60% and 80%. But these values are because of one odd maximum value. Note that in spite of greater time span the random values may have one odd value in the lower range (<10 msec) and this value may cause higher error. To confirm this logic the error in the NT entries was plotted for different categories in Figure 5.23(d). This graph was only plotted for percentages between 60% to 100%. The NT error was categorized as (i) no level achieved for a neighbour in this period (due to message loss etc), (ii) error of up to 1 level and (iii) up to 2 levels. It is evident that no neighbour node has achieved the level error up to 2 in the NT and the effect of message loss is also the same throughout. The higher values were present in the same values as were in Figure 5.21(c) i.e. at 60% and 80% and it is confirmed that they had achieved an error of 1 level. As already explained it was due to the earlier dispatch of some LEVEL beacons. Note that this error may be present for any percentage value but chances are less when a larger percentage of time span was used.

For follow up experiments a value of 90% was chosen as the results were improved compared to other percentages. The 100% is ignored as the nodes are not strictly synchronized and if this value is chosen the values at maximum edges will cause errors.

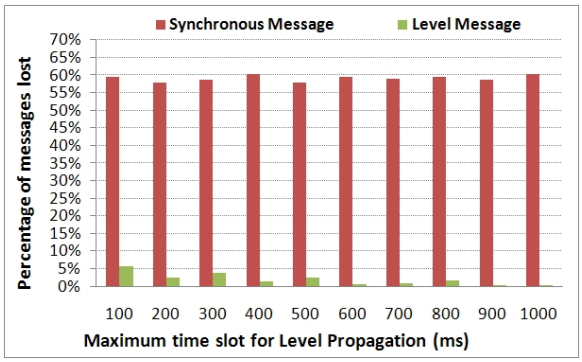
The experiments performed with 500 msec of total time span also gave similar results with some higher values in error percentage coming at different points. When the time was further reduced to half (comparing with 1 second) the errors in the NT and the levels increased.

5.6.11.3 Effect of Level Propagation Delay

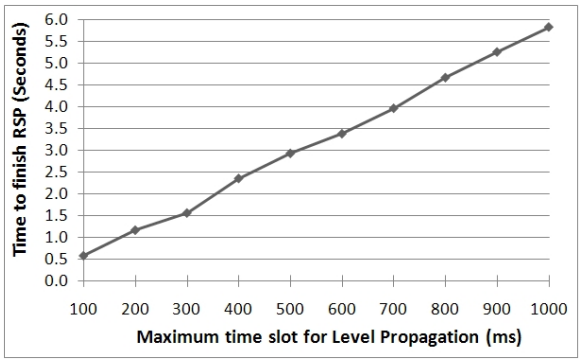
These set of experiments check the effect of *time span for level* in the presence of SYNCHRONOUS messages as compared to in their absence as in Section 5.6.10.2. The only parameter changed from those defined in Table 5.6 is the level propagation delay which is varied from 100 msec to 1000 msec.

Looking at the Figure 5.20(b), time to complete level propagation phase increases with time. This was expected as the time span to broadcast the LEVEL has been increased and obviously will take more time to complete. The percentage of SYNCHRONOUS messages lost is also expected to remain the same because these experiments only effect the LEVEL beacons. The graph in Figure 5.24(a) confirmed this.

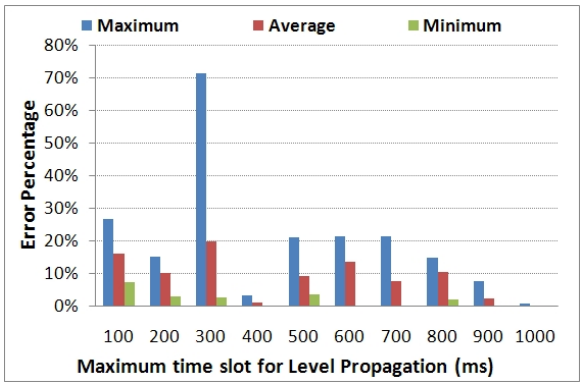
Looking at the Figure 5.24(c), it is obvious that except for the last 2 values (900 msec and 1000 msec) the error percentage varies lot. It was noted that there was a substantial difference between the maximum and the minimum values. At 300 msec there was a very high error



(a) Error percentage in RSP



(b) Setup Time in RSP



(c) Error percentage in node levels and NT entries

Figure 5.24: The effect of Level propagation delay on RSP

percentage and it was noted that for this value there are some entries in the NT that have level error of up to 2. Moreover about 4% of the nodes get level error up to 2. Again it is one off random test that has increased error percentages. For all other cases the error varies between 0% to 20%. Experiments were carried out only 10 times and in some cases the error did not achieve the minimum value of 0% though these remained less than 5%. The results for 100 msec were almost the same as those of higher values but the minimum possible error value of 0% was not achieved even after performing 15 experiments compared to 10 in other cases. The reason being that 100 msec lot of level error (1 level error) was expected as the node clocks were not perfectly synchronized. The error is only reduced when the maximum span of 900 msec and 1 second were used. As the nodes were not perfectly synchronized so the nodes must send levels with as much delay as possible. This is the reason that 1000 msec gave the best results and 900 msec results were better than the previous cases (maximum error of less than 10%). Moreover the 400 msec produces best results (0.22% to 3.24% error). This was only by chance and there was nothing special in 400 msec experiments. A total of 15 instead of 10 experiments were performed for this case but still a maximum error of 3.24% was achieved. The results plotted

for the maximum error in the node levels and the NT level entries in Figure 5.24(c) confirmed that the error is reduced in case of higher time span i.e. 900 msec and 1000 msec.

5.6.11.4 Simulation Results Summary

It was confirmed that by broadcasting a SYNCHRONOUS message by the BS, before the LEVEL message, allows an accurate level assignments if the time span for LEVEL (`TIMELEVEL`) is increased to 1 second. Thus nodes do not need to be perfectly synchronized. Note that after deployment the node's clocks will normally obtain different times as they will start at different times. Initiating the SYNCHRONOUS message also solves this issue. By using one way hash chain the authenticity of message by the BS is also achieved. As this research is not concerned with the encryption issues, so the effect of one way hash chain is simulated by implementing a dummy function which makes the provision rather than the implementation by use of a well defined encryption mechanism.

It was observed that in spite of maximum error creating conditions (high density) the peak error in the node's neighbour and node's level is 1, for 100 node networks, which being too low is negligible. Later it was confirmed that better results can be obtained if utilization time percentage is truncated from both ends rather than from one end (maximum values). This has been renamed to Percentage of time span removed to send from both the ends (`REOMOVEPERCENTTIMELEVEL` defined in software) and it is chosen to be 10% (90-10). This will effectively give 80% of total random values from `TIMELEVEL` but will remove the initial and the final 10% values and helped lot in achieving the correct level. For future experiments `REOMOVEPERCENTTIMELEVEL` term will be used (The percentage of time truncated from edges) instead of *Effect of utilization time*, which was the percentage of `TIMELEVEL` time used. The effect of density, scalability and the noise is revisited in future sections when effect of multiple BSs will be checked. Therefore these are not presently being considered in this section.

5.6.12 Simulation Results with Multiple BSs

In this section the RSP was implemented using more than one BS. It has been done to confirm that RAEED works perfectly under the multiple BSs. For the BS placement and quantity, the steps taken by INSENS were followed. The experiments deploy 1, 2 and 4 BS. The developers of INSENS [19] claim that the best results are achieved when BSs were placed at opposite corners. Therefore the BSs were placed at opposite diagonal corners. The simulation results also include experiments involving 1 BS. As the effect of different factors like Time span for LEVEL (`TIMELEVEL`), Time span for SYNCHRONIZE (`TIMESYNCHRONIZE`) and Percentage of time span removed to send from both the ends (`REOMOVEPERCENTTIMELEVEL`) had already been tested using the single BS and best values were found, therefore these values were taken as constant in these experiments. The test perimeters for these experiments are shown in Table 5.8,

with parameter name, its mostly used value and the range in which that parameter is varied:

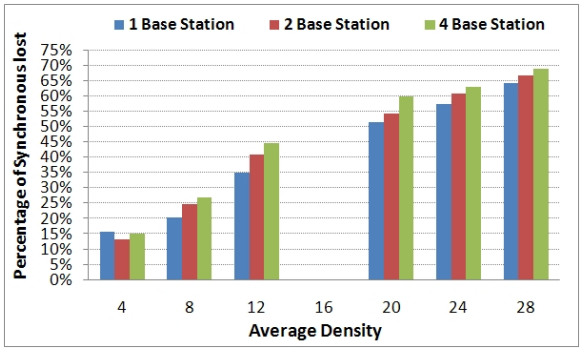
Table 5.8: Parameters used to check the effect of Level propagation delay on RSP

Parameter	**Standard Value**
Number of experiments	5
Number of Nodes	100 and 1000
Number of BSs	1,2 and 4
Node Density	4-28
Noise conditions	no noise
Time span for LEVEL	1 second
Time span for SYNCHRONIZE	10 msec
Percentage of time span removed to send from both ends	10%

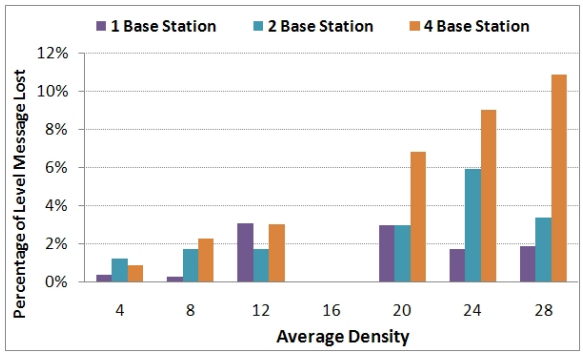
5.6.12.1 Effect of Density

Figure 5.25(c) shows the effect of density on time to complete the RSP using different number of BSs. It is evident that number of BSs had no effect on the setup time. The reason being that all BSs start the RSP simultaneously and as it was a 10x10 grid placement (100 nodes), so all nodes finish nearly at the same time. Also evident from the figure is that the time decreases as the density is increased, the reason being that the node's maximum hop distance from the BSs decreases by increasing the radio range of nodes. Note that to increase the density of network, the placement of nodes remained the same and only the range was increased. As the LEVEL beacons were sent after 1 second so the setup time informs the rough maximum level (hop distance from BSs) of nodes. Maximum level of N roughly means the network will finish the RSP between N and N+ 1 second.

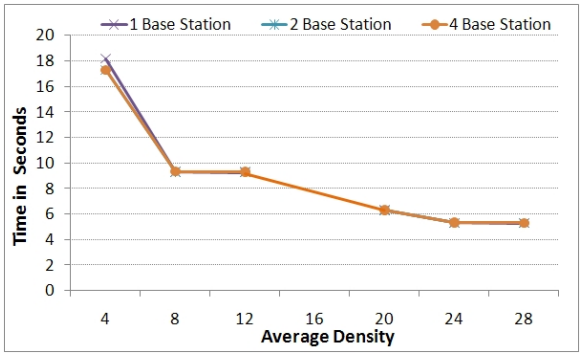
Figure 5.25(a) shows the effect of density on the average percentage of SYNCHRONOUS messages lost in the RSP using different number of BSs. It is obvious that with the increase in density the collisions increased which increased the percentage of messages lost. Except for the case of 4-node density, by increasing the number of BSs, there was an increase of 2% in message loss. The reason being that SYNCHRONOUS messages of different BSs collided at the centre of the network. This is so because all the BSs broadcast SYNCHRONOUS messages at the same time and reach at the centre of network simultaneously. As these collisions were few so their effect was minimal. The thing to note here is that the message lost or the collisions increased to about 70% in case the node density was 28. But as each node contained about 28 neighbours so nodes were eventually synchronized. Even if nodes were not perfectly synchronized then there would be a synchronous error of 10 msec if the nodes receive the SYNCHRONOUS message in next phase and so on. For higher densities this time (`TIMESYNCHRONIZE`) should be increased



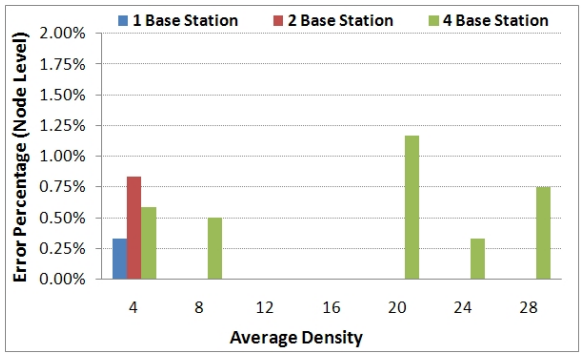
(a) Percentage of SYNCHRONOUS messages lost in RSP



(b) Percentage of LEVEL messages lost in RSP



(c) Setup Time in RSP



(d) Average error percentage in node levels

Figure 5.25: The effect of density on RSP using multiple BSs

to more than 10 msec. But the drawback is that this will induce an error in multiple of more than 10+t msec, where t is the time increase of more than 10 msec e.g. 5 for 15 msec.

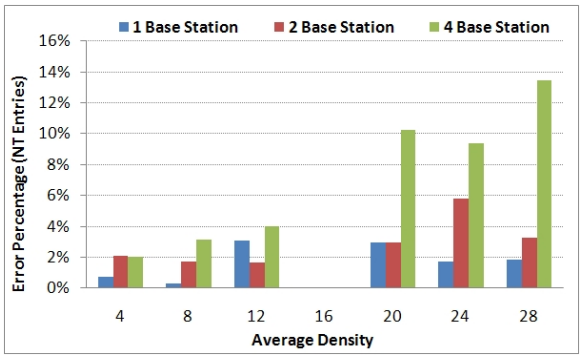
Figure 5.25(b) shows the effect of density on the number of LEVEL messages lost in the RSP using different BSs. Note that the percentage was quite low for 1-BS for all densities and it increased considerably as BSs were increased to 2 and 4 respectively. Remember that the protocol was designed in such a way that the collisions in the LEVEL beacons are minimized because it will assign incorrect levels to nodes and neighbours. But as the BSs broadcast the LEVEL beacons simultaneously, a lot of these will collide in the middle if multiple BSs are deployed. The percentage will increase as the number of BSs are increased. Secondly, by increasing the density, the collisions will increase and so will the percentage of LEVEL message lost. It is rather recommended to broadcast SYNCHRONOUS and LEVEL beacons one by one from each BS, if multiple BSs were deployed. Although this will increase the time to finish RSP with each BS (directly proportional to number of BSs), but will result in the low percentage of message loss compared to the single BS. These experiments will be performed later.

Figure 5.25(d) shows the effect of density on the percentage of error in the node levels using different BSs. It is evident that the error parentage remains less than 1% throughout for all the cases. In 4 density network, the error in node level exists even if 1 and 2 BSs were used. The reason is similar to the previous cases. Loss of a single LEVEL message will induce more error as the number of neighbours is quite low. A node can attain correct level if it receives LEVEL message from any of its lower level neighbours. The collisions will be lesser even in case the density is 4. This is so because of lesser neighbours. The error percentage will not be very high. The reason of error in levels for denser networks, when 4 BSs are deployed, is because of collisions of some LEVEL beacons in the middle as explained earlier. Except for 4 node network the node level error is 0% for 1 and 2 BSs for whatever the density may be. In case of 4 BSs, the error increased up to 1%. These results are encouraging as it indicates that only 1% of nodes will get incorrect levels. It has also been confirmed that this error was up to 1 level for all cases except for 4 neighbour networks.

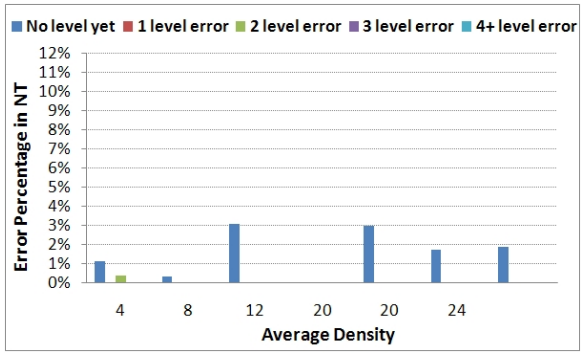
For 4 average density networks the error was always a level of 2 for 1 BS and 2 BSs. While for the 4 BS network the error is split to 0.25% and 0.33% for 1 and 2 level errors respectively. The reason being that the correct LEVEL beacons sent to that node were lost due to collisions. Thus the node gets incorrect level from the next 2 nodes which of course are the higher level nodes in the network tree. Note that in a 4-neighbour grid placement, 2 neighbour nodes are one level higher and the other 2 are one level lower. No two nodes of same level are neighbours. So if the correct level beacons are lost because of noise or collision, the error in levels will always be 2. Moreover when this node will propagate its LEVEL beacon the neighbours will record a wrong level for that node because of propagation of incorrect level. So, same error is expected in NT entries as well which will be discussed later. However the higher level nodes will not assign incorrect levels to themselves. This is so because these get correct LEVEL beacons already arriving from other nodes and nodes accept the best (smallest) level. Hence the error remains small i.e. 1%. If the incorrect node levels were propagated throughout this percentage would have increased. Figure 5.26(a) shows the effect of density on percentage of error in NT levels using different base stations. Note that like the previous case (in Figure 5.25(d)), the error percentage is quite high when 4 BSs are used. This occurred as a result of more collisions. To confirm this effect the error in the level is plotted for different categories for each BS separately in Figure 5.26(b), Figure 5.26(c), and Figure 5.26(d) for 1, 2 and 4 BSs respectively.

Figure 5.26(b) shows that in case of a single BS, the error is mostly because nodes did not attain any level for neighbours. This is due to the loss of message levels because of collisions. For 4-neighbour node network there is however an error of up to 2 levels (0.37%) . The reason for this has already been explained and discussed earlier that in such low density network the error will always be of 2 levels instead of 1 level.

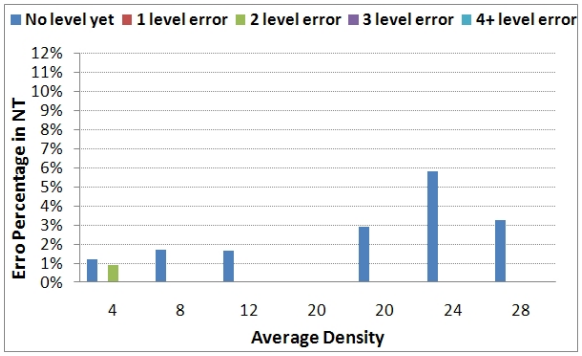
Figure 5.26(c) shows that the results of 2 BS network are similar to that of one BS. Again the error in NT is mostly because nodes did not attain any level for neighbours. This again is



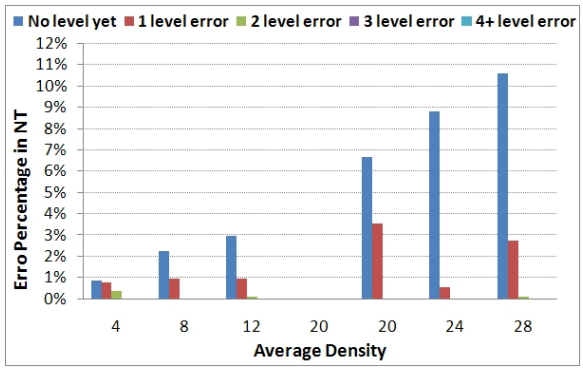
(a) Total Average Error percentage in NT entries



(b) Total Average Error percentage in NT entries using 1 BS



(c) Total Average Error percentage in NT entries using 2 BS



(d) Total Average Error percentage in NT entries using 4 BS

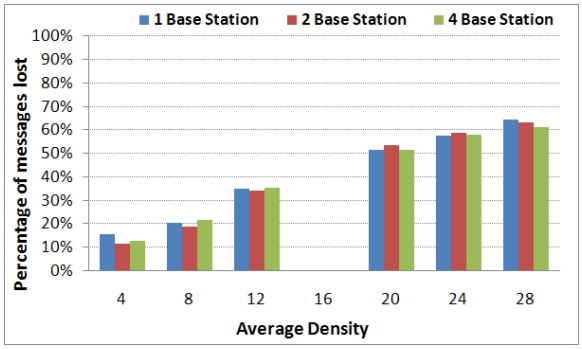
Figure 5.26: The error percentage in NT entries using multiple BS

due to the loss of message levels because of collisions. Again for 4-neighbour node network the error of up to 2 levels (0.92%) is attained.

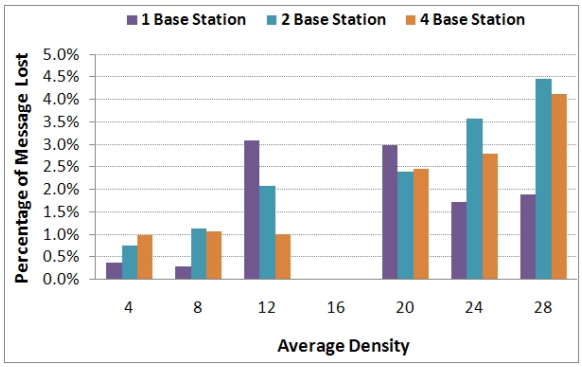
Figure 5.26(d) shows that for 4-BS network the error in NT level entries is mostly due to the loss of LEVEL messages (no level obtained). However 1 level error is also attained in many cases. The reason is the collision in the middle due to 4 level beacons propagated at the same time by different BSs as described earlier. Thus a small percentage (<3% for worst case) of the NT entries attain an error in level up to 1 level. Note that the 4 neighbour networks also attained this error along with the traditional 2 level errors. This was explained while discussing earlier cases.

5.6.12.2 Effect of Density with Slot Time for each BS

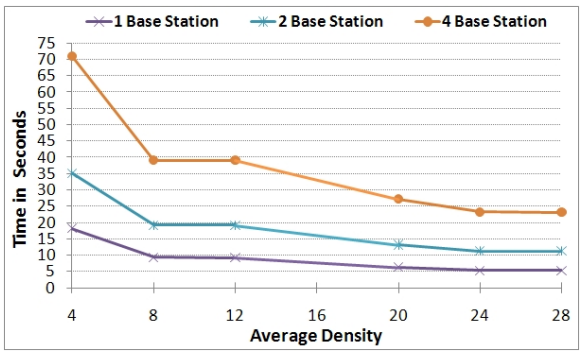
It has been pointed out in the last section that by delaying the level propagation of each BS, the results achieved for the multiple BSs can be improved a lot. In these experiments this approach



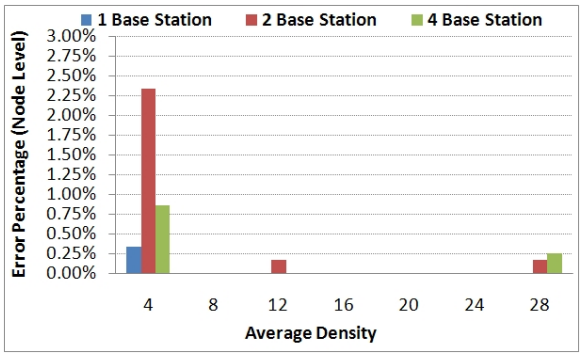
(a) Percentage of SYNCHRONOUS messages lost in RSP



(b) Percentage of LEVEL messages lost in RSP



(c) Setup Time in RSP



(d) Average error percentage in node levels

Figure 5.27: The effect of density on RSP using multiple BSs with independent slot time

was employed. Decision to be taken now was as to how much delay should be introduced. One possible solution is to introduce a fixed delay for all cases. Note that a random delay, without any reason, will also increase the setup time of the RSP. So a logical and compromising delay must be induced. It has earlier been pointed out that the time to complete the RSP depends on maximum level of the node (or maximum hop distance from BSs). So the delay to be introduced depends on the level or hop distance. Repeating the same experiments, as done in previous section on different densities and BSs, the levels were altered for different density networks. By increasing the range, the nodes get closer to the BSs. Thus a delay induced was 1 second more than the maximum level or of same value as shown in the Table 5.9. The Table 5.9 confirms that different delay values are used for each density. Each BS will wait for its turn by that much period of time before initiating the SYNCHRONOUS and LEVEL beacons. So if there are n BSs, the nth BS will send its SYNCHRONOUS beacon n times the delayed time as shown in the table.

Figure 5.27(c) shows the effect of density on the time to complete RSP using different BSs.

Table 5.9: Delay time introduced in multiple BSs for 100 node network

Density	**Max Level**	**Time Value**
4	18	18 seconds
7	9	10 seconds
10	9	10 seconds
16	6	7 seconds
19	5	6 seconds
21	5	6 seconds

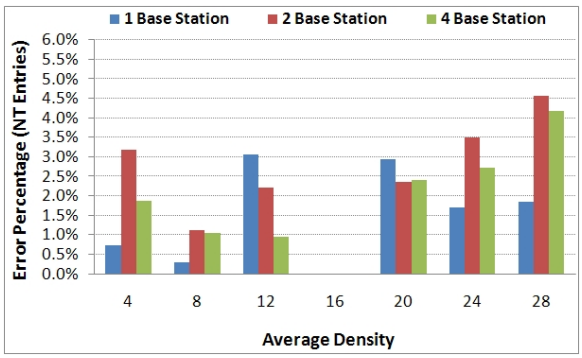
It now confirms that by increasing the number of BSs, the time gets doubled. This is what had already been predicted. The reason being each BS will now take its independent time slot to complete the RSP. So by improving the results the price to be paid is that the network will take more time to complete the RSP.

Figure 5.27(a) shows the effect of density on average percentage of SYNCHRONOUS messages lost in the RSP using different BSs. Comparing it with previous case experiments in Figure 5.25(a), as anticipated,there is no major change. The reason being that nothing had been done to improve the SYNCHRONOUS beacon collisions. The collisions increase as the number of neighbour nodes (density) are increased. Note however that the 2% increase in the message loss per BS has vanished now and the result is almost the same for all number of BSs. This supports our claim in the previous section that it was due to collision of different BS SYNCHRONOUS messages in the middle because this time the BSs sent messages independently in their own time slot.

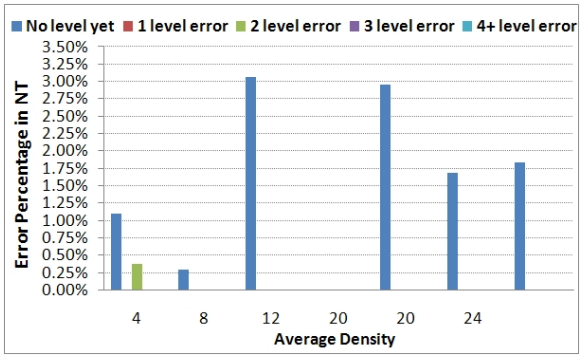
Figure 5.27(b) shows the effect of density on number of LEVEL messages lost in the RSP using different number of BSs. Comparing it with the results in Figure 5.25(b) it is evident that now the percentage of message lost for each BS is almost the same. The little variation in each case is because only 5 experiments have been performed. It is a substantial improvement for 4-BS networks where the percentage of message lost has reduced to one third.

Figure 5.27(d) shows the effect of density on the percentage of error in the node levels using different BSs. Again comparing it with Figure 5.25(d) the results for 4 BSs are much improved now. The error is reduced to 0.25% for 4 BS experiments except for 4 neighbour density network. It has already been debated earlier that due to less number of available neighbours, 4 neighbour networks will give different results. There is a slight increase in error for 2-BS network with 4 neighbour nodes from 1% in Figure 5.25(d) to 2%. But it is anticipated that by experimenting with more than 5 samples, both the results may get closer. Note that due to the unpredictability of 4 neighbour nodes (where message loss plays a major role) the results may still vary with increased number of samples.

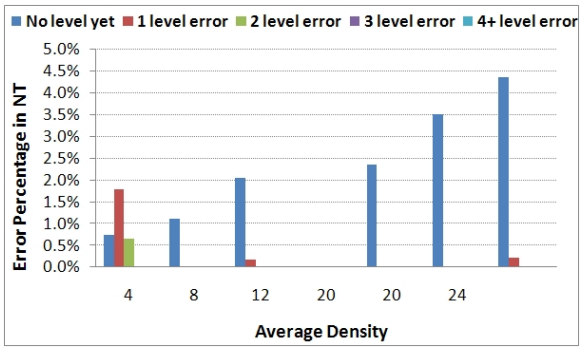
Figure 5.28(a) shows the effect of density on the percentage of error in NT levels using



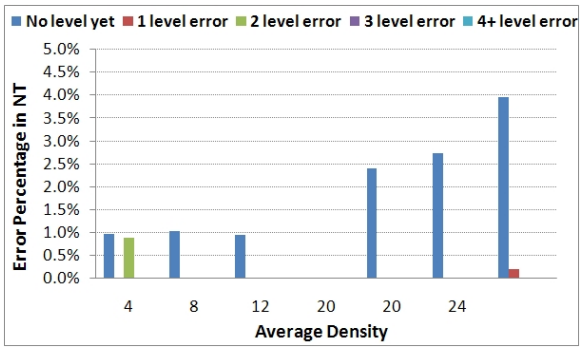
(a) Total Average Error percentage in NT entries



(b) Total Average Error percentage in NT entries



(c) Total Average Error percentage in NT entries



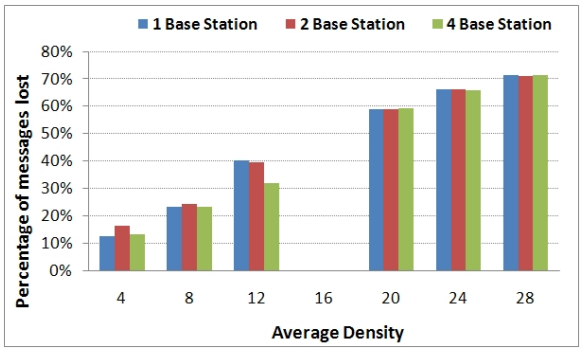
(d) Total Average Error percentage in NT entries

Figure 5.28: The effect of density on RSP error percentage in NT entries using multiple BS with independent slot time

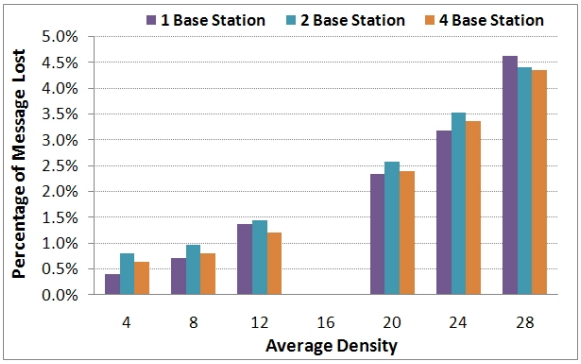
different BSs. Comparing it with Figure 5.26(a), the errors in NT entries have now considerably improved. The maximum error now is about 4% whereas it was touching 14% as displayed in Figure 5.26(a). Also now the error is independent of the number of BSs. While in the previous case as in Figure 5.26(a), the error was dependent on BSs and was increasing with the number of BSs deployed.

Interesting to note here is that the results in Figure 5.28(a) are almost the same as that of Figure 5.27(b) indicating that the error in NT now is only due to the LEVEL message lost (no level attained). To confirm this effect, the error in level is plotted for different categories for each BS separately in Figure 5.28(c), Figure 5.28(c) and Figure 5.28(d) for 1, 2 and 4 BSs respectively.

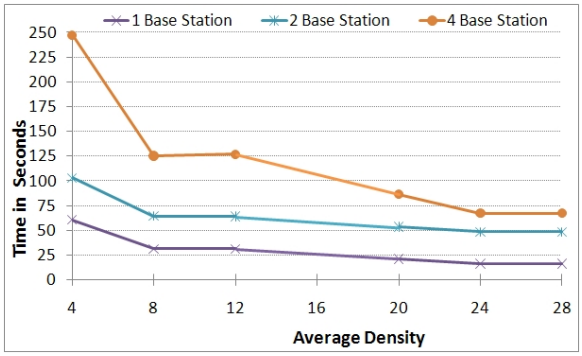
The results for 1 BS are almost the same in Figure 5.26(b) and Figure 5.28(b). The results for 2 BS in Figure 5.28(c) show that like in Figure 5.26(c) , the error in the NT is mostly because nodes did not attain any level for neighbours. This is due to the loss of message levels



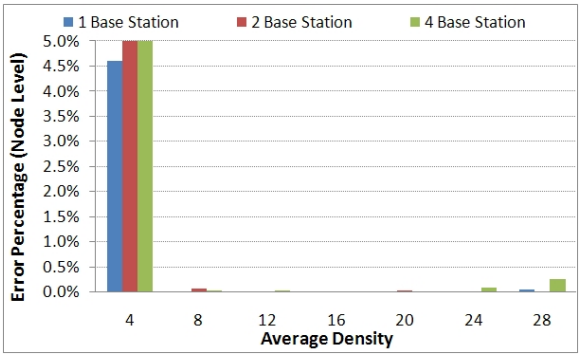
(a) Percentage of SYNCHRONOUS messages lost in RSP



(b) Percentage of LEVEL messages lost in RSP



(c) Setup Time in RSP



(d) Average error percentage in node levels

Figure 5.29: The effect of scalability on RSP using multiple BSs

because of collisions. Again for 4-neighbour node network the error of up to 2 levels (0.92%) is attained. Figure 5.28(d) shows that for 4-BS network most of the error in NT level entries is due to the loss of LEVEL messages (no level obtained) instead of the case as in Figure 5.26(d). Here 1 level error was also obtained for many cases. The reason for this is because of collisions in the middle due to 4 LEVEL beacons propagated at the same time have now been removed, as described earlier. Thus a small percentage (3% for worst case) of NT entries that attained an error in level up to 1 level in the previous experiments in Figure 5.26(d) have now been successfully removed. Note also that the 4-neighbours network still obtained 1 level error along with traditional 2-level error. This has earlier been discussed in detail.

5.6.12.3 Effect of Scalability and Density

These experiments involve scalability check i.e. a 1000 nodes network placed in square grid is checked. The radio range is varied to observe the results for different densities. Again the delay introduced is 1 second more than the maximum level as shown in the Table 5.10:

Table 5.10: Delay time introduced in multiple BSs for 1000 node network

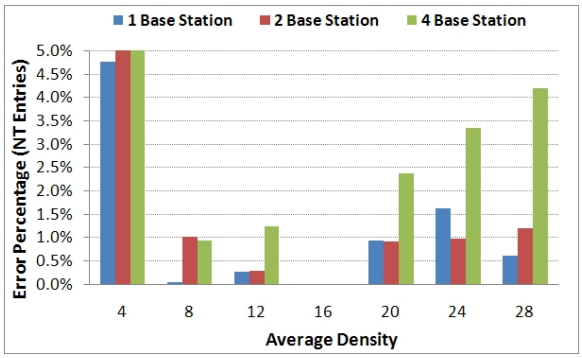
Density	Max. Level	Time Value
4	62	63 seconds
8	31	32 seconds
12	31	32 seconds
20	21	22 seconds
24	16	17 seconds
28	16	17 seconds

Figure 5.29(c) shows the effect of density using 1000 nodes network on time to complete RSP with different number of BSs. It confirmed that the time to complete has increased considerably as compared to 100 node network. The reason being that nodes are placed at greater distance because these are 10 times the number and thus have a higher level as evident in Table 5.10. It had already been indicated that the time to complete is directly proportional to maximum level in the network. Therefore, with increase in maximum level the time to complete also increased.

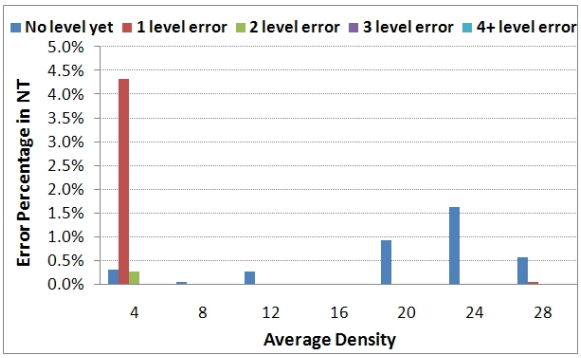
Figure 5.29(a) shows the effect of density on an average percentage of SYNCHRONOUS messages lost in the RSP, using different number of BSs, for 1000 nodes network. Comparing it with 100 node network it is obvious that there is no major change and the results are identical. The reason being that the topology remained constant and only the size of network was increased. Hence, it is obvious that the message loss depends on network topology rather than scalability.

Figure 5.29(b) shows the effect of density on the number of LEVEL messages lost in the RSP through use of different number of BSs, for 1000 nodes network. The results were similar as in 100 nodes network. The message loss increased with increase in density and shot up to maximum of 4%. This confirms that the message loss depends on topology as well as density and not on scalability.

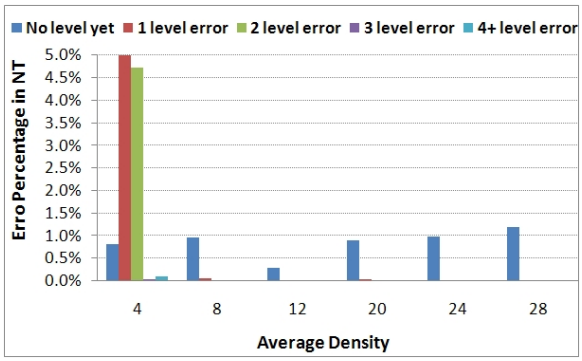
Figure 5.29(d) shows the effect of density on the percentage of error in the node levels by using different number of BSs, for 1000 nodes network. As can be seen, the results are identical as that of 100 nodes except for 4 density network. The error has increased to 4% for one BS network and gone quite high for multiple BS networks. The large value of node error in the case of 4 density node for the multiple BSs (2 and 4 BSs) test has not been plotted properly to enable a better view of the other results. For a clearer view of other values the maximum limit of the graph is set to 5%. The actual values are 31% and 25% for node level errors for 2 and 4 BS networks. This level error also has caused error in the NT entries of 32% and 26% respectively. Further investigation and graph in Figure 5.30(c) as well as Figure 5.30(d) confirmed that the error was mostly minus one level. It was observed that this error was due to the fact that a node in the middle has lost all the SYNCHRONOUS messages and thus was



(a) Total Average Error percentage in NT entries



(b) Total Average Error percentage in NT entries



(c) Total Average Error percentage in NT entries

(d) Total Average Error percentage in NT entries

Figure 5.30: The effect of scalability on RSP error percentage in NT entries using multiple BS

not properly synchronized. This is quite possible in a small density network. When the node receives LEVEL beacon it erroneously assigns itself a lower level. This enables the node to fire LEVEL beacon earlier than when it should have been. Thus whichever node receives that level will also assign itself a lower level and this error spreads throughout. If nodes get a level higher than the actual the error would not propagate because nodes always accept the best level. That is the reason that most of the nodes and the NT entries get many wrong levels of -1. These experiments confirmed that in case of low density networks, the chances of nodes getting incorrect level is high but is still negligible (-1). As already stated, a level error of 1 makes no difference. But this error reduced down to less than 2% as with more neighbours the node is eventually synchronized despite more message loss.

Figure 5.30(a) shows the effect of density on percentage of error in the NT levels using different BSs. The reason for high values in case of 4 neighbours network has already been explained earlier. To confirm this effect the error in level is plotted for different categories for each BS separately in Figure 5.30(b), Figure 5.30(c) and Figure 5.30(d) for 1, 2 and 4

BSs respectively. The graphs again confirm no variation from 100 node network except for 4-neighbour networks.

5.7 Data Forwarding Phase (DFP)

The aim of this phase is to forward data from the source nodes to the BS and pacify the attacks like the gray hole, black hole, jamming, etc. despite the malicious activities like node compromise, node destruction or jammed area. The presented Data Forwarding Phase (DFP) has some resemblance with the Arrive [44] protocol explained earlier in the Section 4.9. However, Arrive has some shortcomings, a few of which have been mentioned by the authors themselves [44] and others identified in this thesis (Section 4.9.3.2). RAEED has attempted to address most of these problems and is thus differs from Arrive. The main differences between RAEED and the Arrive protocol are summarized below:

- Arrive assumes that the nodes already know their level, whereas RAEED uses an authenticated scheme in the RSP to assign levels to each node.
- Arrive acknowledges a defect in the protocol that a malicious node may lie about its level, whereas RAEED uses authenticated level assignment using one way hash chains which prevents a malicious node from successfully lie about its level.
- Passive Participation is employed in Arrive to send redundant data; this has many draw backs. In RAEED all decisions are taken locally and no passive participation is performed.
- Unidirectional links are a problem for Arrive. RAEED removes unidirectional links in the KSP.
- Arrive does not support any encryption mechanism and thus is susceptible to many cryptographic based attacks. The Key Setup Phase (KSP) in the beginning of RAEED assigns each node a pair key for unicast messages and cluster keys for the broadcast. The messages are then accepted only from the nodes which are verified in KSP and possess a correct key.
- Arrive is susceptible to the hello flood and rushing attacks. RAEED removes these threats in the KSP.
- The wormhole attack is not addressed by Arrive and the attack can occur at any time. A black hole attack is always possible in the presence of the wormhole attack. In RAEED an innovative scheme is presented in the RSP (LSP) that rectifies the wormhole attack.
- In Arrive a node may forward a message to oblivion whereas in RAEED it is ensured that a neighbour forwards the data only to a correct two hop node. A Lost signal is generated in the case when no further forwarding is possible.

- The black hole attack is still possible in Arrive, if all parents and neighbours become black holes. This will enable all the higher level nodes attached to that node to lose their data messages. This will also remain undetected throughout. RAEED sends back a Lost beacon thus enabling the node to inform its forwarder that it has been unable to forward the data.

- Arrive employs up to 8 multi-paths, whereas RAEED only selects one path. This reduces the traffic overhead.

This thesis presents two different schemes for data forwarding. The first scheme is named as the handshake scheme (Section A.6.1) and the second is termed as the lost indication scheme. The first scheme involves handshake and multiple paths in data forwarding while the second scheme requires neither of these. The hand shake scheme has some drawbacks including additional message overhead. Moreover, it relies on multiple BSs and multiple paths to work efficiently. This leads to development of a lost indication scheme that has a lower message overhead and avoids handshake, multiple paths, multiple BSs etc. The remaining part of this chapter explains the Lost indication scheme in detail.

5.7.1 DFP Design:Lost Indication Scheme

A node S, upon detecting an event in the environment, generates the data that contains the event or message ID, node's own ID and a source ID (saved in TargetID field). The source node also attaches a data stamp in the HMAC field which is used later by the BS to confirm that the data has not been altered in the relay process by any node (integrity). Moreover the data is encrypted with the node's Independent key (authenticity) so that only the BSs can decrypt what the actual data is (confidentiality). The other nodes just relay/forward this encrypted data without knowing the contents of the data. This provides data authentication, integrity and confidentiality from the source to the destination. The complete message is again encrypted by the source node S using the cluster key K_{C_S}. It is then unicasted to the neighbour node T having a high ranking (reputation). All the neighbours initially have equal ranking. The ranking is based on throughput (number of messages sent versus received) history and the neighbour's level (hop distance from BS). The format of the DATA message is:

$$S \rightarrow T : (DATA, [ID_S, ID_T, MID_m, Stamp_S, ID_S, [datapayload]_{K_{I_S}}]_{K_{C_N}})$$

The target node, upon receiving the data, forwards it to its best ranked neighbour. Here S, T and N are the IDs of the source, the target and the sender nodes respectively:

$$N \rightarrow T : (DATA, [ID_N, ID_T, MID_m, Stamp_S, ID_S, [datapayload]_{K_{I_S}}]_{K_{C_N}})$$

This data forwarding is repeated throughout until the message is received by a node at level 1, which will always forward the data to the BS. If all nodes behave correctly the path

eventually is the best path to the BS. As a follow up some extra parameters like the number of messages forwarded by a node etc can be added to provide load balancing. Note that neighbour nodes can receive the data (as it is broadcast) but they do not forward the data that is not targeted to them. Each node saves the (event, source) pair for some period of time to detect if the same message is received again. It also determines to which neighbours the data has already been sent (eavesdrop or forwarded by the node). This is done to avoid nodes that have already forwarded the same (event,source) pair. After unicasting data to a neighbour, the sender node eavesdrop for a small period, to check has the receiver node forwards the data further. As nodes have knowledge of 2-hop neighbours, so they can also be checked if the data is forwarded to a legitimate node or resulted in oblivion. Thus all possible black hole attackers will be detected. In case a legitimate node does not have any available neighbours to forward the data (i.e. if the event source pair has already been sent to all 1-hop neighbours), a LOST beacon is generated and is sent back in the reverse path. The format of the LOST beacon is:

$$N \rightarrow T : (LOST, [ID_N, ID_T, MID_m, Stamp_S, ID_S]_{K_{P_{(N,T)}}})$$

The data payload can also be placed in a LOST beacon if required. But the (event,source) pair and data stamp are sufficient for the node to infer which data packet has been lost. This also requires the help of a small data base (DB) maintained in each node. This DB is discussed in detail in Section A.7. The node, upon receiving the LOST beacon, attempts once to forward that lost data to another node. Note that this LOST message is only propagated up to one level instead of all the way back to the source. This is to avoid an extra message overhead. The ranking process involved in each node also maintains the number of messages lost by a node (LOST beacon received).

5.7.2 Evaluation of Lost Indication Scheme

5.7.2.1 Formal Verification of Lost Indication Scheme

The hypothesis for this model is that:

"The DFP of RAEED achieves 100% throughput".

It is assumed that all the nodes have already finished KSP and RSP successfully. The **model** is composed of a sink and the node models. The event generator model is not used because the two events, timeout and sense, are generated by the node model. The ***node model*** is shown in Figure 5.31. All nodes start in the LISTEN location. If node is a source (Source flag is set) and all nodes are free (`BusyNodes` is 0), the channel is free (`ChannelBusy` is clear), and the maximum number of messages (`TotalSent`<`MAXMESSAGE`) have not been sent, the node model senses and sends new data (SEND_DATA). A sense message is used to indicate to all nodes that a new data message must/will be sent. In the real world the data is sensed after a fixed time interval and the message IDs are used to indicate the new data. To save state space additional details have

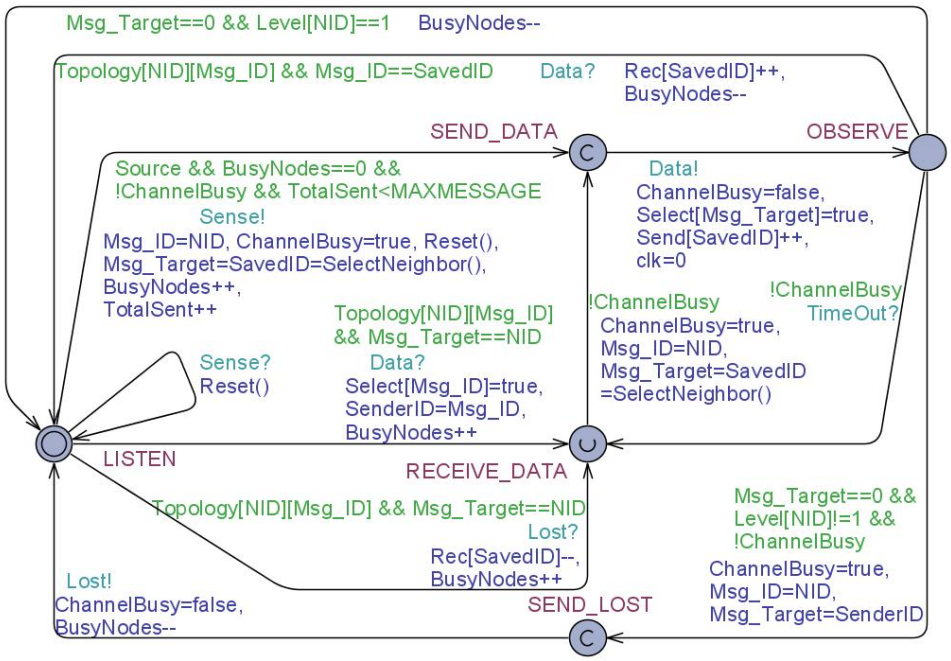


Figure 5.31: Node model used in verification of DFP

been removed. The source node increments the global variable `TotalSent` (indicating the total data packets sent in a network) each time a new data is sensed. The `BusyNode` variable is also incremented as the node model becomes busy and will be free once it goes back to the LISTEN location. The target of a data message is determined using a function `SelectNeighbor()`, that determines the next hop node, depending upon its ranking. The source node then moves to the SEND_DATA location and broadcasts the data. The corresponding `Send[]` variable is incremented to indicate the number of messages sent to a neighbour. It then remains in the OBSERVE location until either the neighbour node to which the data has been forwarded broadcasts the data or a timeout occurs. Note again that, instead of using the clocks, the timeout is modelled by a simple message to save the state space. The node of course will get out of the OBSERVE location straightaway if the data has been sent to the sink. The `Rec[]` variable is incremented in case a neighbour has successfully forwarded the data. In case the neighbour has not forwarded the data, the node moves to the RECEIVE_DATA location and tries to send the data to another node. A flag (`Select []`) is set each time the data is sent to a particular node. This avoids sending the data again to the same node. The flag is also used by the sender node if and when the data is received from another node. These flags are then reset using the `Reset()` function which is executed in all the nodes each time new data is sensed.

If a node, in the OBSERVE location, does not eavesdrop any of its neighbour forwarding data further, a Lost message is generated (SEND_LOST), informing the previous node that, although the current node has received data, further forwarding has failed. This triggers the previous node to transmit the data (FORWARD_DATA) to another node. The current node's ranking is lowered because it did not have any path available.

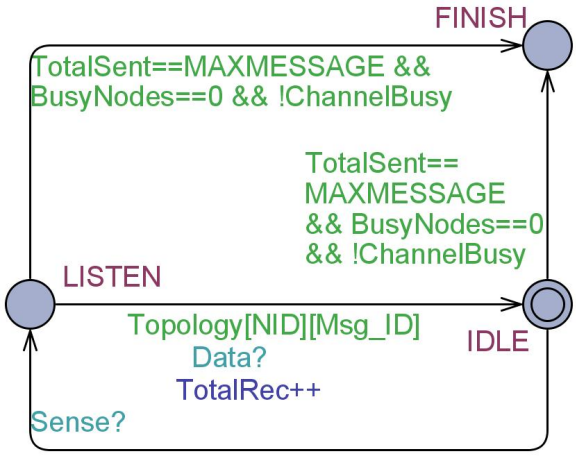


Figure 5.32: Sink model used in verification of DFP

The ***sink model*** is shown in Figure 5.32. The sink model starts in the IDLE location and moves to the LISTEN location after receiving a `Sense` signal. Note that the `Sense` is used as a timeout or as an indication that the new data must be sensed. The sink remains in this location unless either it receives the data (that takes it back to IDLE) or all the nodes, including the source node, have finished forwarding the data (which takes it to the FINISH location). Note that the sink model moves to the IDLE location upon receiving the data and increments the global `TotalRec` variable (indicating the total number of data packets received). This is done so that the variable is not incremented more than once in a single sense event. Note also that, when multiple paths are used, the sink model can receive multiple copies of the same data. But the variable should be incremented only once. In case the maximum number of messages allowed have been sensed (`TotalSent` is equal to `MAXMESSAGE`), the channel becomes free and all the nodes are in their idle location (`BusyNodes` is 0), the sink model then moves to the FINISH location and the model deadlocks here.

The **Verification** process involves verifying the following claims/properties:

1. The data is transmitted fairly
2. The nodes after broadcasting the data never deadlock
3. No deadlock in the RAEED protocol
4. The source node will always send the desired number of data messages
5. The source node will always receive the desired number of data messages
6. The throughput of the data messages is always 100%

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.2.5. All the claims were proved for a variety of topologies confirming that the DFP has been implemented successfully with 100% throughput.

5.7.2.2 Simulation Results

Computer simulation were undertaken to confirm the working of the DFP. It was observed that the throughput remained unaffected for different sizes and densities of networks. For these experiments, each individual node acted as a source and broadcast data only once. A single BS was used and always placed at the corner. Ten experiments were performed under noise free conditions and the average throughput received at the BS computed. The density was varied for a 100 node network placed in a 10x10 grid. A throughput of 100% was observed for all cases. For scalability testing the networks of 100, 200, 500 and 1000 nodes were placed both in grid and random placement. It was observed that the throughput remained unaffected. Another important parameter to note was that the average number of hops taken by the data message to reach the BS. Because there is neither an attacker nor noise, data followed the shortest path without needing any LOST beacon or regenerating on any of data packet more than once. The average number of hops also indicated that the best path was selected for the data forwarding by all the nodes.

5.7.3 Effect of Noise on DFP

This section evaluates the DFP, or in fact the complete protocol, under noisy conditions. The protocol is designed in such a manner that it should withstand message loss due to noise. The methods involved for evaluation are formal modelling and computer simulation.

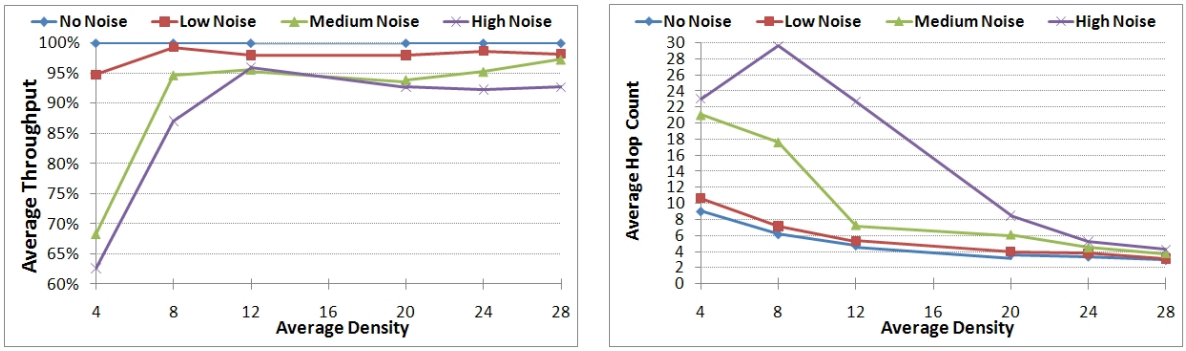
5.7.3.1 Formal Verification

The model used in the Section 5.7.2.1 was reemployed but modified to accommodate the RF noise (message loss), as was done in the Section 5.5.4. The threshold value for RF noise was varied between 10% to 40% and it confirmed that all the properties presented in the Section 5.7.2.1 are satisfied including the data transport property which has confirmed 100% throughput. The model was checked for all possible 5 node networks and grid networks of node size 9 and 16 were also checked. The data transport property confirmed that the throughput remained unchanged despite the presence of noise.

5.7.3.2 Simulation Results

To quantify the results and to test larger networks, computer simulation were undertaken to confirm the effect of noise on the DFP. There noise samples as indicated in Section A.4.3 and Section 5.5.4 were reused. A 100 node network, in a 10x10 square grid, was used with a single BS at the corner. Node density was varied by increasing the radio range of the nodes.

Figure 5.33(a) shows the effect of noise on the data throughput. The throughput remained unchanged (100%) in the absence of any noise and despite the presence of the hidden terminal problem. Except for the lower density networks (4 & 8), the throughput remained above 90%



(a) Effect on data throughput

(b) Effect on hop counts taken

Figure 5.33: The effect of noise on the DFP

for all the noise samples. The slight decrease in throughput was because of the messages loss due to the RF noise at the BS. Note that the messages lost at the BS will not be detected by the neighbour nodes as the BS is a sink and does not forward the data messages further. However, any data loss before reaching the BS will enable the sender node to resend the data. The data, thus, eventually reaches the BS. The only cost paid is the redundant messages, i.e. a greater number of hops the data will travel instead of following the ideal path.

A notable results is low throughput when a medium/high noise sample is employed in a low density (4) network. This is because of two reasons (i) excessive loss of data message and (ii) fewer available paths. Each node can have a maximum of 4 neighbours with one node being the message sender. Sometimes because of the message loss, the route takes the data to the border nodes which will not be able to forward it further (has one more neighbour to which the data can be forwarded and this could be the border node as well). The graphs indicate a low throughput data percentage and the high average number of hops taken in this case. It is thus proved that the protocol attempts to find the path for all cases. However, in presence of high noise sample and extremely low density (4) the throughput is quite low. For the 8 node density networks, the throughput is increased as the available paths have increased. Finally, for the higher density networks the throughput is above 90% and as indicated, this loss of throughput is due to the RF noise at the BS. The protocol avoids selecting the same neighbour again for particular data forwarding. Hence this is one of the reasons that throughput suffers in the presence of fewer available neighbours (4 & 8). The results showed that RAEED supports high throughput even under noisy environments.

Figure 5.33(b) shows the average hop count taken in data forwarding, showing that this increases by a factor of 3 for lower densities in the presence of medium/high noise. The neighbourhood scheme suffers, as the nodes eavesdropping messages after forwarding data, may not receive the feedback. Some of these messages are lost because of noise. The nodes, thus, re-

send the data if the feedback data message is lost because the data has been considered lost. Thus sometimes data is received multiple times via multiple paths. This value is highest when the routes are the longest and settles down as the density is increased thus enabling smaller paths (nodes become near to the BS). Note that the low unexpected value for a 4 node density network, as compared to 8 node density, in the presence of high noise is because most of the messages are lost (60% throughput). The average hop count approaches the ideal (shortest path) value as the density is increased. Note that for average density of 24 and 28 nodes, the hop count is almost the same, in the presence of all the noise samples.

This is one of the shortcomings present in RAEED, that in the presence of high noise, the data can be sent multiple times (message overhead). This drawback exists for other noise samples but becomes more pronounced and substantial in the presence of high noise sample. However, note that the same overhead is present if multiple paths are employed for message passing. In case of multiple paths the overhead will exist in the presence of all the noise samples and even in the absence of noise. On the contrary, in RAEED this overhead is prominent only in the presence of high noise. Moreover, this overhead enables high throughput as compared to the multiple paths schemes (INSENS) in the presence of high noise. Another reason why the average hop count increases because of noise is that a lot of LEVEL messages are also lost which enables the nodes to get incorrect levels for themselves and for their neighbours. An error in the KSP and the RSP will always be inherited by the DFP, thus enabling a lower throughput and higher average hop counts. Finally, because of the lower ranking of the nodes, in cases where messages are lost due to noise, the nodes are ignored the next time. Thus, an alternative path is taken for data routing.

The protocol is proved to be robust against the noise but the only drawback is that the noise near the BS, should be low. This assumption may be realistic because the BS is in a safe place and it is the more distant nodes that are usually deployed in hostile conditions. Moreover this problem is solved by adding an ACK beacon sent by the BS when it receives the data. This message will only travel one hop and it is similar to the FAIL beacon:

$$B \rightarrow N : (ACK, [ID_B, ID_N, MID_m, Stamp_S, ID_S]_{K_{I_N}})$$

After employing the ACK message it was confirmed that throughput recovered to 100% for higher densities. The problem still persists with low density networks in the presence of high/medium noise.

5.8 Chapter Summary

This chapter presents a new routing protocol design that can be a potential solution against many DoS attacks. The protocol design has been rigorously tested using formal modelling to remove the hidden bugs/errors. Computer simulation were performed later to support and

quantify the results as well as to check the effects of scalability and node density. The complete protocol is divided into 3 main phases KSP, RSP and DFP. The KSP involves exchange of keys and bidirectional verification. As the protocol assumes that an encryption mechanism is already in existence so the KSP of the INSENS and the LEAP protocols are used as an example. It was also proved that the KSP presented in RAEED is more efficient as it provides the same security properties and has a lower message overhead. This lower overhead is because the keys are piggy bagged in the bidirectional verification phase and a three way message exchange is performed which reduces a lot of data traffic. Finally the effect of noise is also monitored on the KSP using formal modelling and simulation.

The RSP involves authentic level assignment to each node, which is the hop distance from the BS. The nodes also perform neighbour ID exchange and loud test to verify 2-hop neighbours. This is done to remove any virtual connection between the nodes. The authenticity is provided using one way hash chain known only to the BS. However, only the effect of hash chain is modelled and details are avoided. Each sub phase is explained and tested separately. The RSP is extensively tested using both formal modelling and the simulation.

The DFP involves propagation of data from the source to the sink. Node reputation and level is used in the data routing. The nodes are ranked based on their performance by eavesdropping their activities after forwarding the data. The scheme uses a single path and BS. A lost message is generated in case a legitimate node cannot forward the data further. This enables the predecessor node to forward the data on another path. The scheme was evaluated using the formal modelling and computer simulation. Finally, the effect of noise on DFP was evaluated.

Chapter 6

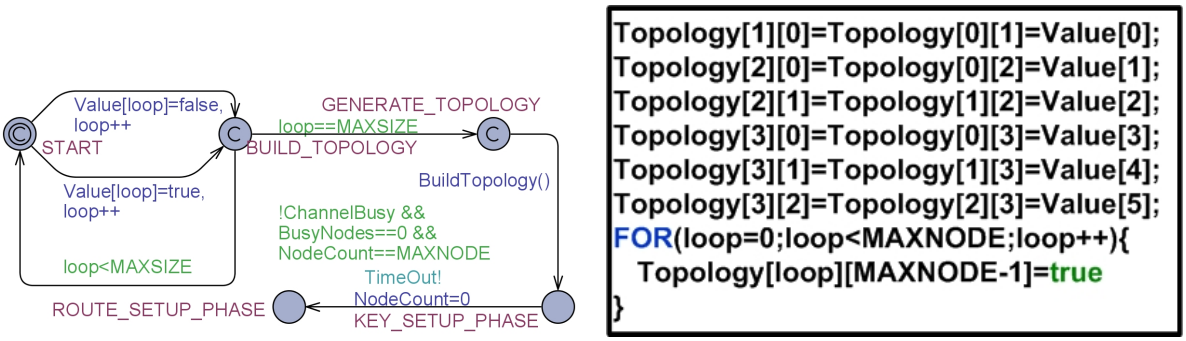
Evaluation of RAEED Against DoS Attacks

6.1 Introduction

This chapter evaluates the new protocol, RAEED, against different DoS attacks. The thesis has already rigorously checked some published routing protocols against DoS attacks in Chapter 4. The method adopted was formal modelling and the results were than supported using computer simulation. The same approach is adopted in this chapter to evaluate RAEED. The rest of this chapter is organized as follows: The attacks rectified by RAEED are explained in Section 6.2 (hello flood), Section 6.3 (wormhole and INA), Section 6.4 (sinkhole), Section 6.5 (tunnel attack), Section 6.6 (rushing attack), Section 6.7 (black hole), Section 6.8(gray hole) and Section 6.9 (jamming) respectively. Finally, a summary of chapter is presented in Section 6.10.

6.2 Prevention Against the Hello Flood Attack

It has already been proven (using formal modelling) that the INSENS and LEAP protocols are immune to the hello flood attack because of its bidirectional verification. As the KSP in RAEED is a modified version of LEAP/INSENS and the protocol employs the bidirectional verification, it is indeed expected to be immune from the hello flood attack. Thus, formal modelling and computer simulation are employed to confirm that the hello flood fails in RAEED. The results are later supported by the practical implementation. Note that, unlike INSENS, this work does not claim that solving the hello flood also solves the rushing attack. It has been indicated in Chapter 4 that the rushing attack is possible in the presence of INA and wormhole attacks. Therefore this thesis will only assert that RAEED will resist the rushing attack once it is proved that it can avoid the hello flood attack, INA and wormhole attack.



(a) Event generator model

(b) BuildTopology() function

Figure 6.1: Models used in the hello flood attack solution

6.2.1 Formal Verification

The hypothesis for this model is :

"The hello flood attack is unsuccessful and the unidirectional links are always avoided in RAEED".

It is assumed that the hello flood attacker is very powerful and has a radio transmission range that covers all the nodes. Moreover, in this model, all possible topologies of network of up to N nodes are checked using a function `BuildTopology()` which simply converts the array `Value[]` into the `Topology` matrix as was done for the Flooding protocol (Section 4.4.2).

In performing the formal verification to confirm that the protocol is immune from the hello flood attack, the node model that was employed earlier in Section 5.5.2.1 was reemployed. For this model it has already been verified, that in the absence of any attacker, all the neighbouring nodes are updated correctly. The requirement is to prove the properties presented in Section 5.5.2.2 in the presence of a hello flood attacker. The formal framework employed in Section 5.5.2.1 requires all the possible N topologies to be tested one by one. This time, however, the model checks for all the possible N topologies automatically. This modification is performed by amending the *event generator* part of the model. The new event generator model is shown in Figure 6.1(a). In previous cases this modification was not adopted because of the state space explosion problem. In this case, however, the model being small (only the KSP has been modelled), all topologies can be checked. The claim to prove that hello flood is unsuccessful involves proving following properties:

$$A[](Problems == 0)$$

$$Protocol.ROUTE\_SETUP\_PHASE \Rightarrow (Problems == 0)$$

Both the properties proved true, confirming that the hello flood attacker was not able to add

itself to a single node in all possible N node network topologies. Note that `Problems` is a global variable that is incremented if a verified neighbour is found to be different from that in the connection matrix. Up to 6 node networks were checked with one node being the attacker and the remaining 5 nodes forming different topologies. It is worth noting that Uppaal checked all 1024 node symmetric link topologies automatically and confirmed that the nodes were immune from the hello flood attack for all those topologies. Only topologies where all nodes possessed bidirectional links were checked in the presence of a hello flood attacker. Note that a hello flood attacker is actually a unidirectional link that must be avoided and the examination of asymmetric links for 5 nodes would involve checking 1,049,000 topologies. There is no advantage in checking all those topologies when it has been confirmed that the protocol ignores nodes with asymmetric links. Furthermore, the formal model was limited to 5 legitimate nodes because by increasing it to 6 symmetric links one has to check 32,768 different topologies. This will cause a state space explosion problem when the automatic topology generator is used and it is not feasible when topologies are employed manually (one at a time). To further confirm the results all the possible 4 node networks were checked using the formal framework in the absence of any attacker. It was confirmed that the unidirectional links were not accepted as legitimate neighbours.

6.2.2 Computer Simulation

The hello flood attacker was implemented using TOSSIM and 100 and 1000 node grid networks were checked for different densities. Two types of hello flood attackers were employed:

- One hello flood attacker that has unidirectional links with all the legitimate nodes. Thus all network nodes can receive the message transmitted by this intruder.
- Four hello flood attackers placed at the four corners of the grid such that each attacker's message can be picked up by 25% of network nodes.

After applying both types of the hello flood attackers on 100 and 1000 node networks, it was confirmed that the attackers had not been added as neighbour by any node. All the nodes marked the hello flood attackers as unverified neighbours in the NT.

6.2.3 Practical Implementation

Finally, RAEED's KSP was implemented in hardware using MICAZ motes [233]. Different networks up to 10 nodes were implemented and checked. Each node, upon completion of KSP, displayed its neighbour's ID using the LEDs confirming that nodes always detected the legitimate neighbours. It was also observed that neighbours placed far apart, having fading signals, sometimes did not add to one another. Sometimes one node adds a neighbour in such conditions but others reject it because of unidirectional link. This, therefore, supports that the unidirectional links are avoided in the KSP.

Finally a 10 node network with a hello flood attacker was also placed in such a way that the attacker can be heard by all legitimate nodes. It was observed that all the nodes did not add the hello flood attacker as their verified neighbour. These experiments confirmed that the hello flood attack presented in RAEED is also practicable. These experiments also proved that nodes in the protocol always reject unidirectional links.

6.3 Prevention Against the INA and Wormhole Attack

6.3.1 Traditional INA and Wormhole Attack

It has been previously shown using the formal framework (Chapter 4), that both INSENS and LEAP are susceptible to the wormhole attack in the KSP. RAEED is also vulnerable to the wormhole attack in the KSP and RSP if the wormhole attacker is already present in the deployment area. This attack will also be possible when new nodes are later deployed. The chance of a successful wormhole attack at the deployment stage is small as the attackers must know a node's frequency. Moreover, as the KSP is finished very quickly so any deployment of the wormhole after KSP will have no affect. The reason being that the nodes will reject the tunneled messages as they will be from unverified nodes. But as the chances of wormhole still exist so an innovative solution for wormhole is presented in this section that with an assumption that nodes can transmit a more powerful message if required after the RSP. This will enable a node to transmit message to a node two hops away, which in case of a normal transmission is not within the radio range of a node. This assumption is realistic as a node's transmission usually can be adjusted at the run time without adding any extra hardware.

If the current KSP is adopted, the chances of a successful wormhole attack at the deployment stage are low as the attacker must know node's frequency. Moreover, the KSP finishes very fast and any deployment of the wormhole after the KSP will not have any effect because nodes will reject the tunnelled messages as they will be from unverified nodes. As the chances of wormhole still exist, an innovative solution for wormhole is presented in this section to pacify the INA and wormhole attacks. This solution is universal and thus can be adopted by any other routing protocol to avoid these attacks. This will also enable the KSP to be modified if required in future work because the scheme does not rely on the KSP.

As stated earlier in Section 5.6.4, the scheme assumes that nodes can transmit a powerful message to a node two hops away. The nodes have already acquired information about their 1-hop and 2-hop neighbour node's IDs after the Neighbour Propagation Phase (NPP). Therefore, without a wormhole tunnel or INA between the nodes, the high power messages, through use of a high power transmission, must reach nodes which are actually located within two hops. A node unicasts a high power LOUD beacon to each independent 2-hop node or may send a single broadcast LOUD beacon using high power. Although, the unicast technique may induce more traffic in the network (thus requiring more energy and overhead); it will enable the nodes

to transmit messages using the cluster key of that 2-hop neighbour. It will also enable nodes to delete the global key before this phase. If the current KSP is adopted, deleting their global key as early as possible is very important (for immunity from node capture attacks). When a node is captured by a wormhole attacker, it might enable the other tunnel adversary to generate LOUD signals similar to this scheme. A solution for such a case is presented later in Section 6.3.4.

However, it is worth noting that whenever a wormhole attack takes place, it is always assumed that the attacker did not know anything about encryption. In this context, this assumption gives flexibility to the scheme presented in this thesis. Note the assumption in this context that the global key cannot be captured and effort to finish the KSP within 2-3 seconds will also prevent an adversary from capturing the nodes before launching the wormhole attack. In future, if a new KSP or a different cryptography scheme has to be adopted, care must be taken to adopt this loud scheme as soon as possible to avoid nodes being captured. The preferred sequence is to perform bidirectional verification followed by the wormhole test and then perform the key exchange. But again this should be done only if wormhole or INA is assumed to be launched after node capture. This, up to our current knowledge, has not been a characteristic recognized in the wormhole/INA. These attacks are assumed to be both powerful and cheap because they did not depend on cryptography.

6.3.2 Formal Verification

The hypothesis for this model is that:

"The INA and wormhole attack are unsuccessful in RAEED".

In addition to the assumptions stated in Section 4.3.3, it is assumed that the wormhole tunnel is at least 4 hops. The wormhole attack is likely to span many hops to cause maximum damage, it is assumed that the 2-hop nodes are closer to the current node compared to the tunneled node. Moreover, it is assumed that the legitimate nodes can increase their transmission power.

6.3.2.1 Model

The formal framework used 3 types of messages:

- The '`Event`' is an event occurring in the system and is generated by an event generator. This event could be elapse of time, an occurrence in the environment, etc.

- The '`Message`' represents the messages exchanged between the nodes. The `Topology` matrix is used to determine which nodes will receive a message sent by a particular node in the network. The message's fields are expressed using global variables starting with `Msg_`. The RF power of a message is modelled using a global flag `Msg_Power` with the value true indicating transmission with the high power and normal transmission otherwise.

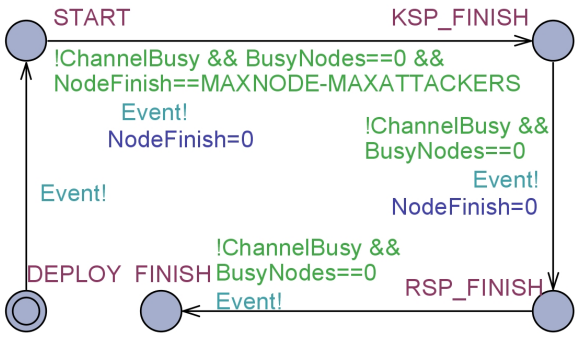
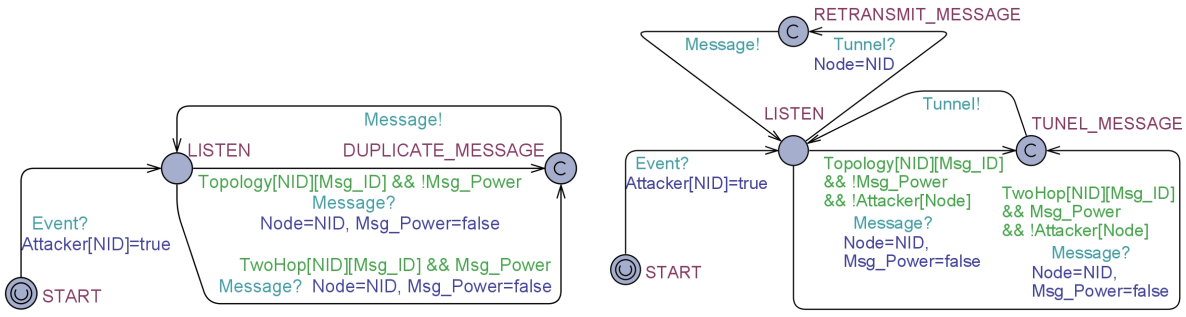


Figure 6.2: Event generator model used in the wormhole solution



(a) INA Model

(b) Wormhole Model

Figure 6.3: Attacker models

- The '`Tunnel`' uses a different RF frequency or is a wired connection used by the wormhole nodes. This message can only be read by attackers; the legitimate nodes know nothing of it.

The formal model to test wormhole attack and INA consists of an event generator, an attacker and multiple node models. The ***event generator model*** is shown in Figure 6.2 and starts in a DEPLOY location. This model starts all nodes by triggering an *Event* message and moves to the START location. It waits in this location using a guard that checks that all the nodes go through the KSP (`FinsihNodes` variable has value `MAXNODE`) and no node is busy handling any received message (`BusyNodes` becomes 0). The model then triggers all nodes and itself to move to the KSP_FINISH location. Later the model moves to the RSP_FINISH location and finally to the FINISH location by checking if all the node models become idle (`BusyNodes` becomes 0).

The ***attacker models*** are shown in Figure 6.3(a) and Figure 6.3(b) for INA and wormhole attacks respectively. Both attacker models moves out of the START location using the

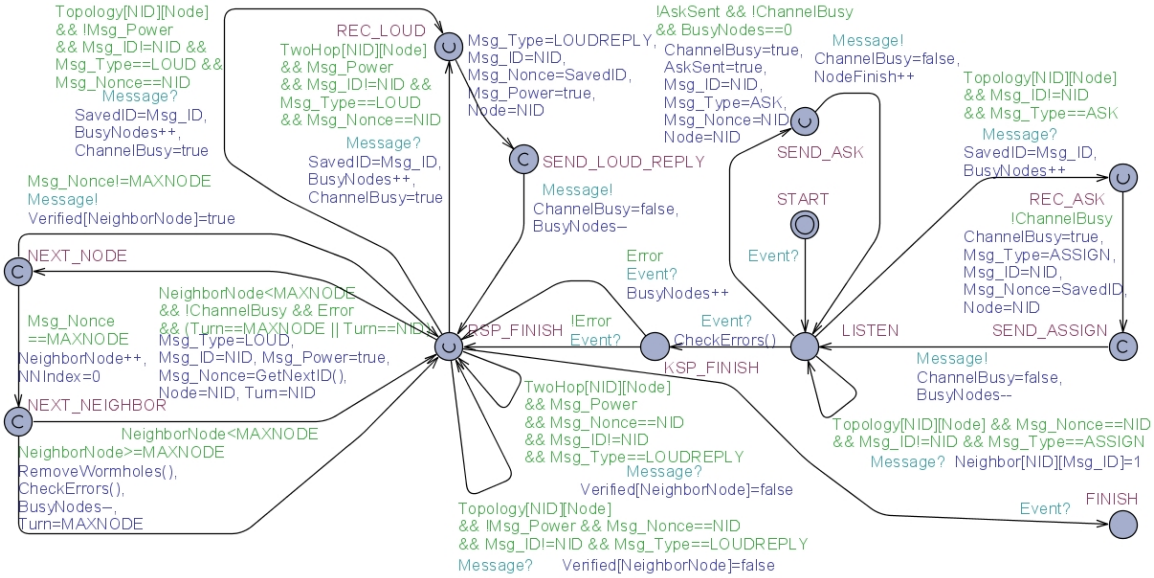


Figure 6.4: Node model to check the wormhole attack

event generator's `Event` signal and the `Attacker` flag is set to indicate that the current node is an attacker. It then remains in the LISTEN location and eavesdrops messages. If any message is received the INA simply retransmits this message and goes back to the LISTEN location; whereas the wormhole attacker tunnels the message to another wormhole attacker node (TUNEL_MESSAGE). The wormhole attacker, upon receiving a tunnel message, retransmits it (RETRANSMIT_MESSAGE). Note that the attacker node remains hidden throughout and does not take part in the KSP. It can also receive a high power message but treats it in a similar way.

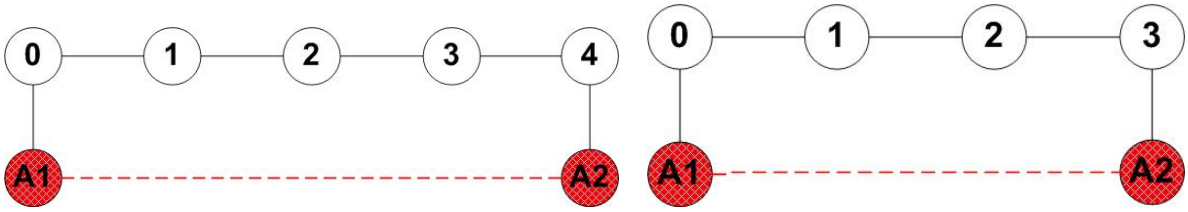
The ***node model***, shown in Figure 6.4 starts in a START location and moves out of it when triggered by the `Event` generated by event generator. It then remains in the LISTEN location during the entire Key Setup Phase (KSP). The KSP includes send ASK beacon (SEND_ASK), receive ASK (REC_ASK) and send ASSIGN (SEND_ASSIGN) locations. The node sends the ASK beacon as it is turned on and expects a reception of the ASSIGN beacon in response. This two way handshake provides the bidirectional verification solution. This models the simplified version of the KSP in the protocol. A self transition in the LISTEN location provides the receive ASSIGN operation and the corresponding `Neighbor` flag for the receiver node is set to true, indicating that the neighbour has passed the bidirectional verification phase.

Once all nodes have finished their KSP, the model moves to the KSP_FINISH location. The node model then checks, using a function `CheckErrors()`, if there is any error in the

neighbourhood information, i.e. existence of any neighbour in the node's memory that is not part of `Topology` matrix. Factually this test is not possible, so in the real world a node will assume that an error is present in its neighbourhood information and would perform the wormhole attack check test. However, for INA this test is feasible because in the presence of a INA, a node always receives back its own messages therefore INA is always detectable. In the presence of a wormhole attack a node usually has more neighbours than are expected so this can be used as a indicator in real world networks to perform the wormhole attack check. Therefore, a node can detect a wormhole attack, in most cases, and INA, in all cases. In the model, however, the `Topology` matrix is used to check this error to reduce the state space. This test was performed only for nodes that have incorrect node IDs in their neighbourhood list. The model also skips the Neighbour Propagation Phase of the protocol and assumes that when the model moves from the KSP_FINISH to the RSP_FINISH location this task has been performed, again to save state space. As the `Neighbor[]` array (models NT in a node) is global, so nodes can automatically check their 2-hop neighbour nodes without performing message passing in the model. In real situations each node would broadcast its neighbour information protected by the cluster key.

A little modification is performed here to enable each node to check its 2-hop neighbours first before allowing the other nodes to perform the same process. A variable `Turn` is used to indicate which node has its turn and is used in the wormhole detection so that one node is checked at a time. Note that initially `Turn` has the value `MAXNODE` and when a node starts processing its 2-hop nodes, the variable is updated to a node's ID (`NID`). It is then reset to `MAXNODE` again after a node checks all the neighbours for a wormhole. The node uses a function `GetNextID()` to get next valid 2-hop nodes and thus sends a LOUD message to it. If all neighbours (`NNIndex`) of the current neighbour (`NeighborNode`) have been checked the variable `NNIndex` is reset to 0 (NEXT_NODE) and goes back to the RSP_FINISH location. The function returns a value that is stored in the `Msg_Nonce` variable. If the value is valid (not equal to `MAXNODE`), the node transmits a LOUD message with high power; a flag `Msg_Power` is used to indicate that the message is transmitted with a high or low power. Moreover the `Topology` matrix contains 1-hop nodes and the `TwoHop` matrix contains 2-hop nodes. A flag, `Verified`, is set when sending a LOUD beacon to 2-hop nodes for a particular neighbour and is cleared if a LOUDREPLY is received from that 2-hop node (self transitions in RSP_FINISH location). This process is done to save state space as in the real world the messages are sent randomly. If a random transmission is used at the modelling process, the node model would have to be modified to add more locations with no additional advantage.

Upon receiving the LOUD beacon (REC_LOUD) from a 1-hop or 2-hop neighbour, the node replies back with the LOUDREPLY using high power (SEND_LOUD_REPLY). When a node verifies all its neighbour's neighbours by sending them the LOUD beacon, one by one, a function `RemoveWormholes()` removes the neighbour nodes for which their verified flag is still



(a) A topology in which wormhole has failed using loud scheme

(b) A topology in which formal framework detected wormhole is successful

Figure 6.5: Topologies used in the wormhole attack

high. This means that the 2-hop nodes connected to them are not reachable and thus these nodes are neighbours via a wormhole tunnel. When all nodes finish this wormhole attack check, the event generator moves the node models to their FINISH locations.

6.3.2.2 Verification

In this formal model some of the properties require node IDs to determine if these nodes have detected the wormhole or not. The nodes next to a wormhole should detect the wormhole while the nodes outside the radio range of wormhole should not detect wormholes. For these properties an example topology is shown in the figure 6.5(a). The white coloured nodes are the legitimate ones with their IDs shown. The red coloured nodes are the malicious nodes with the red link showing the tunnel. Note that for INA the properties remain the same but IDs will change depending on the topology. In order to check whether the system operates as desired the following claims/properties must be proved:

1. The ASK beacon is received by legitimate neighbours
2. The protocol will finish the KSP
3. The protocol will finish RSP
4. No deadlock in the system
5. No legitimate node is declared as a wormhole
6. All the wormhole tunnels are detected
7. All wormhole tunnels are removed by the LOUD scheme

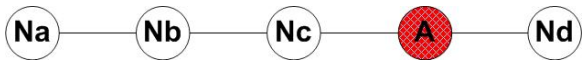


Figure 6.6: One of the topologies used in a hardware implementation of INA

8. All the nodes record their correct neighbours after the LOUD scheme

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.3.1. All the claims including the safety (Claim 5,6,7) and the liveness checks (Claim 4,8) were proved. This confirms that the wormhole attack and INA fails in RAEED.

6.3.2.3 A Successful Wormhole Attack

It was observed that there is possibility that the scheme may not detect the wormhole attacks in certain topologies (Figure 6.5(b)). A reason for such failure is that the tunnel is too short (contradicting the assumptions earlier on). When nodes 0 and 3 exchange LOUD messages they are received by nodes 2 and 1, respectively, since they are the only 2-hop neighbours. Note that even a single neighbour at either end of the tunnel which is not a 2-hop neighbour of other nodes can detect the presence of the wormhole attacks and thus can remove it.

The formal model has confirmed that the only case, in which INA or wormhole attacks are not detected (and removed), is when the nodes next to these virtual nodes do not have a single neighbour pair of nodes which are outside the range of two hops. It is worthwhile to note that this type of network is highly unlikely (normally nodes have many neighbours) so one can safely say that the scheme proposed will be able to detect wormhole attacks in most cases. Moreover, the wormhole tunnel is normally far apart (spans many hops) to give maximum damage. This further reduces the chances of getting such a topology in network (3 hop limit). Therefore, it will be a rare case that such a topology may exist to fail the proposed loud scheme.

6.3.3 Practical Implementation

A number of topologies were implemented with 5 and 6 nodes in hardware using MICAz motes [233]. This was carried out to confirm that the LOUD scheme can check the effect of INA in different topologies. One such topology is shown in Figure 6.6. However this work needs extensive refinements and more details. Also, more hardware experiments would have to

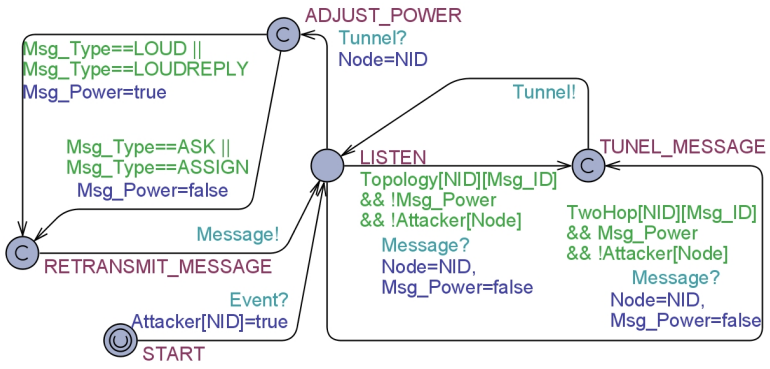


Figure 6.7: Wormhole Attacker Model

be performed to examine environmental affects like radio fading etc.

6.3.4 Intelligent Wormhole/INA (Encryption Failure)

It was stated earlier that if the intruder launches INA or wormhole attack after overcoming encryption, it can fail this newly presented scheme. This is because the attacker can observe the nodes sending the LOUD beacon by decrypting the messages and can also broadcast the LOUD beacons with the higher power as well, thus jeopardizing the effectiveness of this scheme. It was also stated earlier that such attacks have not been characteristic of a wormhole or INA intruders because these attacks are launched without the help of cryptographic failure. However, as this kind of attack may be launched in the future, so a solution for that scheme is presented in this section.

The solution to this attack is to unicast the multiple LOUD beacons with the variable power. Only individual nodes will know whether the LOUD beacon is transmitted with the high power or not. The received LOUDREPLY thus can determine the presence of wormhole or INA. An INA or wormhole attacker will be in uncertain here as what to do. If it transmits all the LOUD beacons with powerful signal to reach 2-hop neighbours, the legitimate node will receive unexpected 2-hop neighbour's LOUDREPLY even when low power LOUD is sent. If it does not transmit LOUD with high power the wormhole/INA is detected straightaway. Thus a wormhole or INA can easily be detected even if the intruder utilizes knowledge of cryptographic details. The only drawback being that extra LOUD beacons will consume more energy in nodes.

6.3.4.1 Model

The modified wormhole ***attacker model*** is shown in Figure 6.7. The attacker is an improved version of the one shown in Figure 6.3(b). Whenever it receives a tunnelled LOUD or

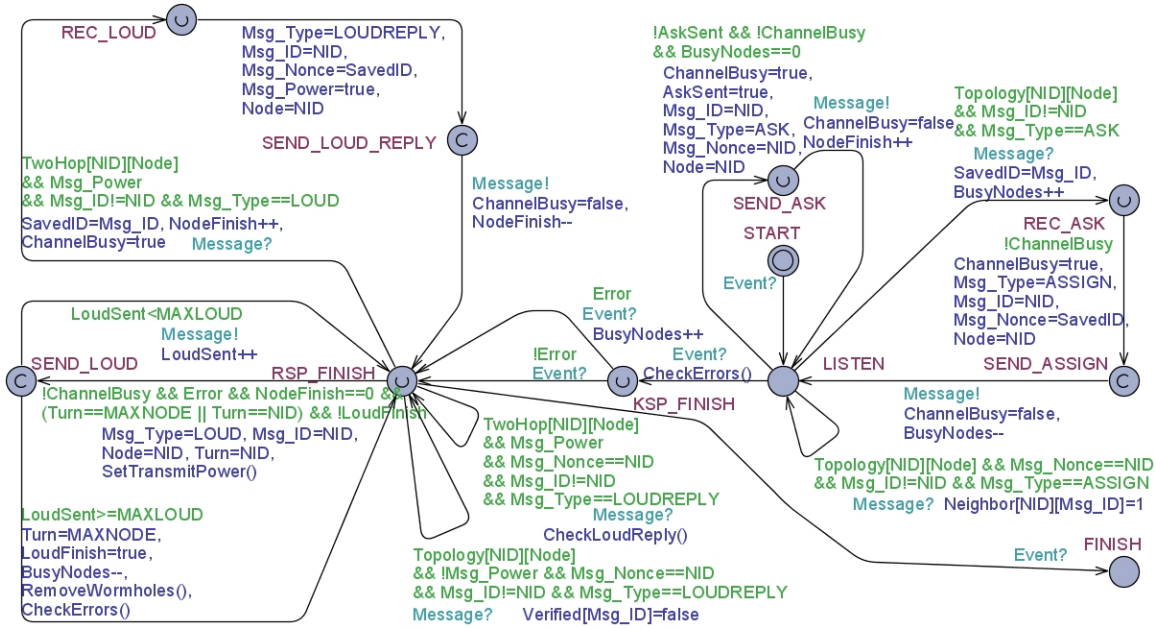


Figure 6.8: Node model to check the wormhole attack

LOUDREPLY message, it rebroadcast it using a high power at the other end. It was observed that such a wormhole will fail the initial scheme as mentioned in Section 6.3.1 (the properties mentioned in Section 6.3.2.2 failed).

To avoid such an attack the node model is modified as mentioned earlier. The modified node model is shown in Figure 6.8. The model is simplified after the RSP_FINISH location as compared to the one in Figure 6.4 because unnecessary details have been removed. Moreover, the LOUD beacon is broadcasted instead of being unicasted to individual neighbours, further simplifying the model. The LOUD beacon is sent `MAXLOUD` times instead of once and only sender node has the knowledge that the LOUD beacon is a fake (normal power) or original (high power). The received response (LOUDREPLY) is then evaluated using the `CheckLoudReply`() the function. A 2-hop neighbour, responding back with the LOUDREPLY when a LOUD beacon is sent with normal power, is declared as a wormhole link. A flag `TestFail` is used to indicate such a neighbour. In the `RemoveWormholes`() function any neighbour having either the `Verified` or the `TestFail` flag set is removed from neighbour list.

6.3.4.2 Verification

All the properties in Section 6.3.2.2 have been proved as true in presence of the new node model indicating that the scheme is not vulnerable to the wormhole attack/INA even if the attacker uses the cryptographic information in rebroadcasting the tunnelled message. The properties in

claims 5 and 6 have been modified now because the flag `TestFail` is used to indicate a wormhole is detected by the node model. The modified claims are described again in Section B.3.2. Both the claims were proved true confirming that the wormhole fails even if encryption is inefficient.

6.3.5 Intelligent Wormhole Attacker with Signal Detection

This scheme has a flaw if both the wormhole attackers can detect the signal strength of all the nodes within their range and transmit message at the same RF power as received messages. If a message received by one end of the wormhole has a high RSSI value (i.e., sent with high power), this end notifies the other end of the wormhole to also transmit the relayed message at high power. This will allow the 2-hop neighbours to receive this message. An attacker node can use the higher transmission power when it receives a LOUD message, therefore the nodes that are two hops away from the end of the wormhole will receive this message and reply back. This becomes evident when the attacker model is allowed to detect signal strength by giving access to the `Msg_Power` flag and the attacker then transmits messages depending on received signal's strength. In such a case the properties will fail. Note that the attacker in this case does not even have to overcome the encryption as mentioned in Section 6.3.4.

However, detecting signal strength accurately to confirm that the message is sent by a node with high or low signal is very difficult to implement, especially when nodes are scattered randomly. It also adds a delay in message received via virtual links due to process involved in signal measurement etc. But as the possibility does exist, some wormhole and INA connections may persist after the loud scheme has finished. RAEED however solves this issue in the later phase as the virtual link will only be continuously used for data forwards if it behaves correctly i.e. forwarding each data message that it receives. Otherwise the neighbourhood watch scheme adopted by the protocol will detect the misbehaving links and will transmit data via another link. To prove this all possible 5 node topologies were checked in the presence of one virtual link as was done in the formal framework earlier for other protocols. The model used was the one presented for DFP in Section 5.7.2.1 with ACK sent by the BS to 1-hop nodes. It was confirmed that all the virtual links (INA/wormhole) had failed to prevent data routed by the source node to reach the sinks. This confirms that RAEED eventually will fail INA and wormhole attack no matter how intelligent the attacker is. To our best knowledge, this solution using formal modelling to overcome the wormhole attack and INA has not been presented before. Nor does there exist a solution that works in resource constrained WSN nodes without any extra cost. More experiments however should be performed on hardware implementations of this new scheme under different environmental conditions. This has been left the subject of future work.

6.4 Prevention Against the Sinkhole Attack

In order to confirm if RAEED is immune from the sinkhole attack, the model used to verify functionality of the RSP in Section 5.6.9 is reused; this model has already verified the functionality of SPP and LSP, each part of RSP. The sinkhole is only possible in these two sub-phases of the RSP. The other two possibilities of the sinkhole, through the use of hello flood or wormhole have been removed in Section 6.2 and Section 6.3. Thus only possibility remaining is if the intruder enables the legitimate nodes to believe that it is a BS.

The one-way hash chain is used to calculate the MAC, attached in the message. The nodes and the attacker can only calculate if the MAC is correct but cannot generate the MAC; only the BS is able to generate the MAC. Thus the possible attacks for the sinkhole attacker is to attach a spoofed (fake) MAC or replay an old MAC received by the BS. Another intelligent sinkhole attack is an attempt to disturb the routing process by becoming attractive to other legitimate nodes. This can be achieved by broadcasting the LEVEL beacon immediately instead of after the usual time delay. Therefore, the important thing to verify is how much maximum damage it can inflict. For the protocol to run without any side effect, like message overhead etc, the maximum damage should be within 1 level error even if the sinkhole has a hidden link with the BS. This section verifies the effect of a sinkhole attack by employing formal modelling and computer simulation.

6.4.1 Formal Modelling

The hypothesis for this model is:

"RAEED is immune from the sinkhole attack".

Apart from the assumptions stated in Section 4.3.3, it is further assumed that the time tick is in milliseconds and errors less than 1 ms are ignored. Moreover, apart from all 5 node topologies, a single 9 node grid topology is also checked. It is also assumed that the one way hash chain employed by the BS for authentication cannot be decrypted by an attacker.

6.4.1.1 Model

The model is similar to that presented in Section 5.6.9. Some modifications are made, however, for additional details required for the sinkhole attack prevention. First the message authentication code is added to the messages. Second, in order to determine which nodes in the network can receive the message in the channel, `Msg_ID` is not used in the `Topology` matrix. The reason being that the attacker can replay old messages in this model and when attacker transmits replayed messages `Msg_ID` will not be the ID of an attacker. So, persisting with `Msg_ID` in the channel will become a cause of error in the message reception/transmission. An extra global variable, `Sender`, is used instead, which is adjusted before each message transmission. This has been added to allow the correct working of the radio channel. A third addition is that the

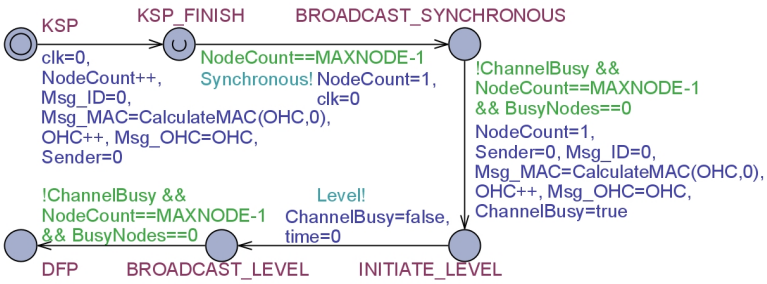


Figure 6.9: Sink model to check the sinkhole attack

nodes, upon receiving any message, check if the sender node is present in its NT entries. In RAEED, the nodes always verify whether the sender node is a verified neighbour and that the cluster key is valid. This test was missing in the last model (Section 5.6.9) because there was no attacker and model was simplified. Instead of adding an extra field for the cluster key, the model confirms the sender node's verification by looking at the `Topology` matrix for the sender node. This check employs `Msg_ID`, which is the message sender ID present in actual message. Thus each legitimate node, upon receiving any message, has an additional guard of `Topology` matrix using `Msg_ID`. The modified models are shown in Figure 6.9 and Figure 6.10 for sink and node model respectively.

The ***attacker*** model used for sinkhole is shown in Figure 6.11. It is obvious that the attacker does not follow the protocol specifications. It can transmit fake LEVEL or SYNCHRONOUS beacons at anytime, just after the KSP is finished. In order to save state space the attacker undergoes a single attempt to broadcast the two messages independently. Note, however, that the Uppaal tool will automatically check for all the possible cases. Thus an attacker's attempt at any possible time is checked automatically. The attacker also records the legitimate messages and replays them.

6.4.1.2 Verification

The verification involves all the properties verified earlier in Section 5.6.9. Moreover, the following additional claims were checked:

1. Attacker's SYNCHRONOUS messages will always be rejected
2. Legitimate node's SYNCHRONOUS messages will always be accepted
3. Attacker's LEVEL messages will always be rejected

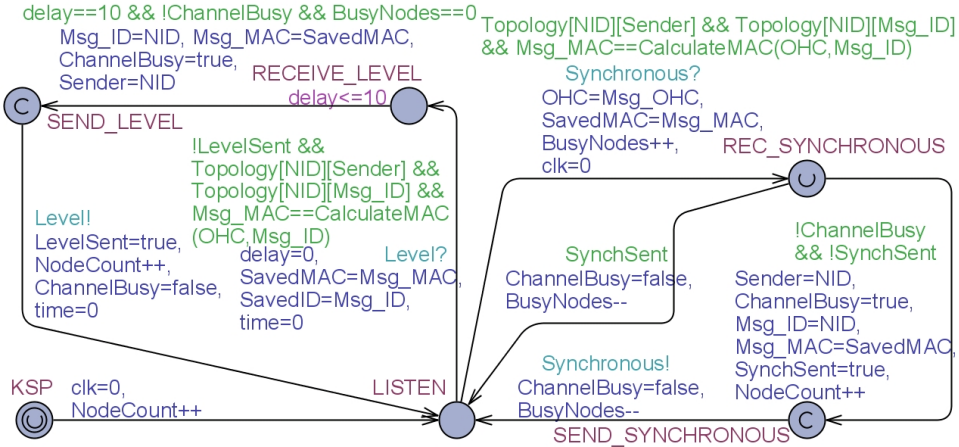


Figure 6.10: Node model to check the sinkhole attack

4. Legitimate node's LEVEL messages will always be accepted

A detailed description of the above claims in terms of Uppaal properties is explained in Section B.3.3. The proof of the new claims confirmed that the sinkhole attacker fails to become attractive by sending fake messages. The claims presented in Section 5.6.9 were also proved true confirming that replay attack by the sinkhole to behave as a BS had also failed. RAEED with the help of the encryption technique thus resists the sinkhole attack.

6.4.2 Computer Simulation

As stated earlier, this research does not evaluate encryption. However, using the property of the one-way hash chain, that messages can be decoded by nodes but cannot be encoded, the encryption was modelled in the simulation. The sinkhole attacker cannot create an authentic MAC generated by the BS. But it can replay or generate fake messages. It was confirmed that the sinkhole fails in RAEED.

6.5 Prevention Against the Tunnel Attack

It has been indicated earlier that the current research considers the wormhole attack different from the tunnel attack; in the wormhole attack the attacker remains invisible, while in tunnel attack the attackers are the compromised nodes of the network which are also connected via hidden tunnel (wired or through different frequency wireless link). If the tunnel attack becomes successful in the RSP, the attackers becomes attractive enabling the sinkhole attack as well.

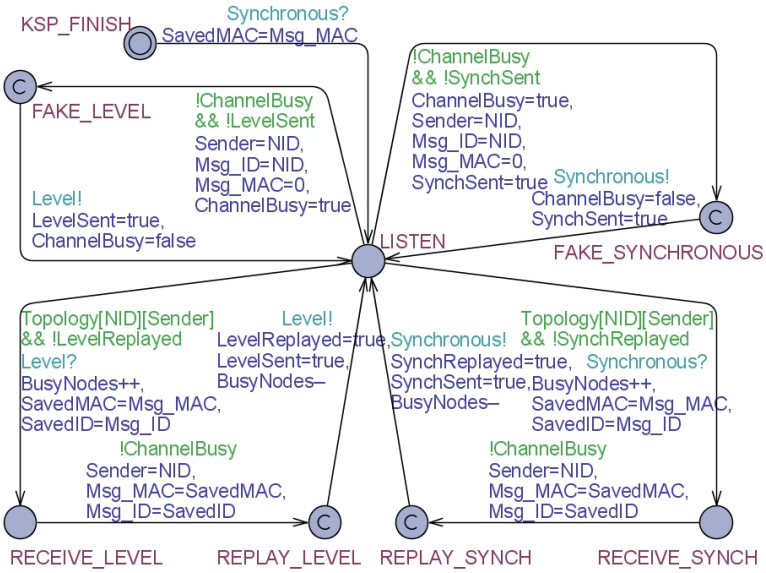


Figure 6.11: Attacker model to check the sinkhole attack

The rushing attack is also possible as the legitimate LEVEL beacons will be later rejected. In order to totally resist the sinkhole attack, it is necessary to avoid the tunnel attack.

6.5.1 Formal Modelling

The hypothesis for this model is that:

"RAEED is immune from the Tunnel attack".

The assumptions used in sink hole attack evaluation (Section 6.4) have been retained. Also it is assumed that the BVP of KSP and NPP of RSP have successfully completed.

The ***model*** used is similar to that presented in Section 5.6.9 which was later extended in Section 6.4.1. More modifications are made, however, for additional details required in the models to avoid the tunnel attack. The Source field, used in the LEVEL beacon to indicate from which node the LEVEL beacon was received, is added in the message format. The additional check then upon receiving the LEVEL beacon is to confirm that the Source is indeed a legitimate 2-hop neighbour. Note that the 2-hop neighbour's information is gathered in the NPP and the model assumes this phase has finished. In the node model a third guard is thus added on reception of a LEVEL beacon (RECEIVE_LEVEL) to confirm the 2-hop test. The same check can also be added for the Synchronous messages. But as the attacker gains no advantage in tunnelling those messages, the test is not added in the model or the protocol. Note that Synchronous beacons are flooded in the network and are transmitted as soon as these are received. So their earlier dispatch does not give any advantage to that intruder. The modified models are shown in Figure 6.12 and Figure 6.13 for the sink and the node models respectively.

The ***attacker*** model for tunnel attack is shown in Figure 6.14. The model starts in the KSP location and then moves to the LISTEN location. Upon receiving the LEVEL or the SYNCHRONOUS message it tunnels this message to its other counterpart attacker model

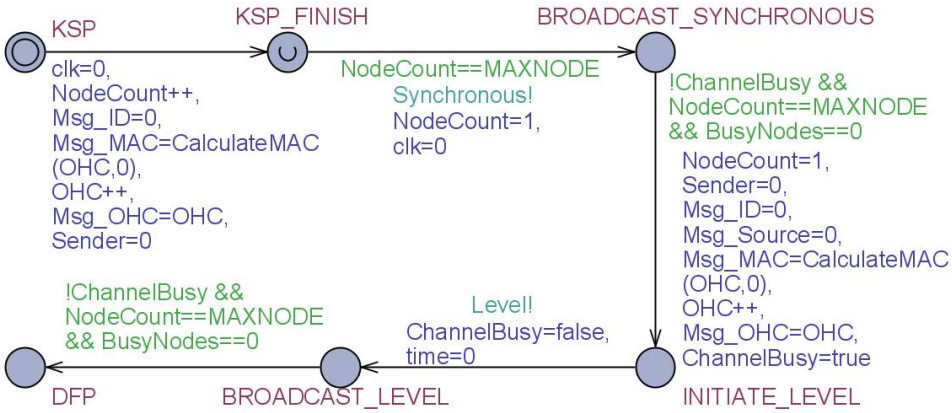


Figure 6.12: Sink model to check the tunnel attack

using a *Tunnel* message. Note that this message is only known to the attacker and is missing in the node or sink model. Upon receiving a tunnelled message, the attacker broadcasts the same message to other legitimate nodes. The message remains unchanged and the sender ID is replaced to enable legitimate nodes to accept this data.

The ***verification*** involves all the properties verified earlier in Section 5.6.9. The verification results indicate that the tunnel attacker fails in RAEED as the nodes reject the tunnel level and the attacker fails to persuade any node to assign an incorrect label.

6.5.2 Computer Simulation

Computer simulation were later performed to check the effect of a tunnel attack on larger networks. A 100 node grid network was chosen first and experiments were performed for different densities. Two tunnel attackers were placed, first one near the BS and the second one near the opposite corner of BS. It was observed that the tunnel attackers did not have any effect on the results and nodes got their correct levels with a maximum error of up to 2 levels. Later, the effect of scalability was also checked and a 1000 node grid network was tested for different densities. The results again confirmed that the tunnel attacker fails in RAEED.

6.5.3 Tunnel Attack in Combination with Framing Attack

It has been shown that the tunnel attack fails in RAEED. However, if a tunnel attacker is designed in a more complex way, and the framing attack is associated with it, the current protocol might fail to avoid the tunnel attack. Although this form of the tunnel attacker, to our knowledge, has not been presented the current research explains how by combining these two

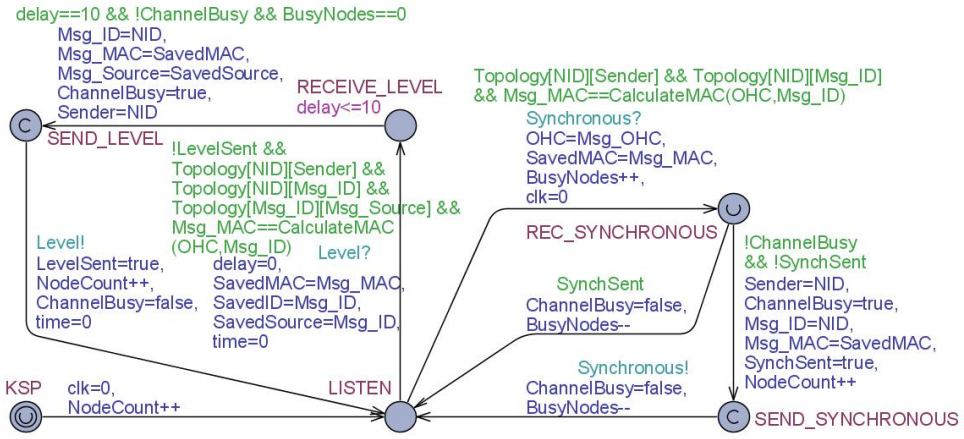


Figure 6.13: Node model to check the tunnel attack

attacks, a tunnel attacker can be successful. The intruder, by realizing that the legitimate nodes check the 2-hop neighbour before accepting the LEVEL beacon, can add a spoofed legitimate 2-hop neighbour ID in the message. The legitimate nodes now will accept a tunnelled message in the current scheme. This has been proved through modelling a new attacker model and the properties presented in Section 5.6.9 about node levels now failed. It therefore confirms that the tunnel attack is successful in this modified form. So, in order to avoid this kind of tunnel attack, a solution is presented which utilizes the loud messages. A node, upon hearing that another neighbour node has framed it by sending a fake LEVEL beacon, transmits a loud beacon similar to as presented in Section 6.3. Thus all the 2-hop neighbours which may possibly have received the tunnelled message will reject the message. The message format is:

$$N \rightarrow * : (LOUD, [ID_N, ID_F, nonce, -, -]_{K_C})$$

Here F is the ID of the node which has sent a spoofed LEVEL beacon in the name of node N and K_C is the cluster key of the sender node N. This additional LOUD message is added in the node model which is shown in Figure 6.15. The properties presented in Section 5.6.9 about node levels are verified and were now proved correct. This confirmed that the modified tunnel attack has also been rectified by RAEED.

6.6 Prevention Against the Rushing Attack

The only way a rushing attack is possible in RAEED is during the RSP by propagating SYNCHRONOUS or LEVEL messages in advance. As SYNCHRONOUS messages are flooded and

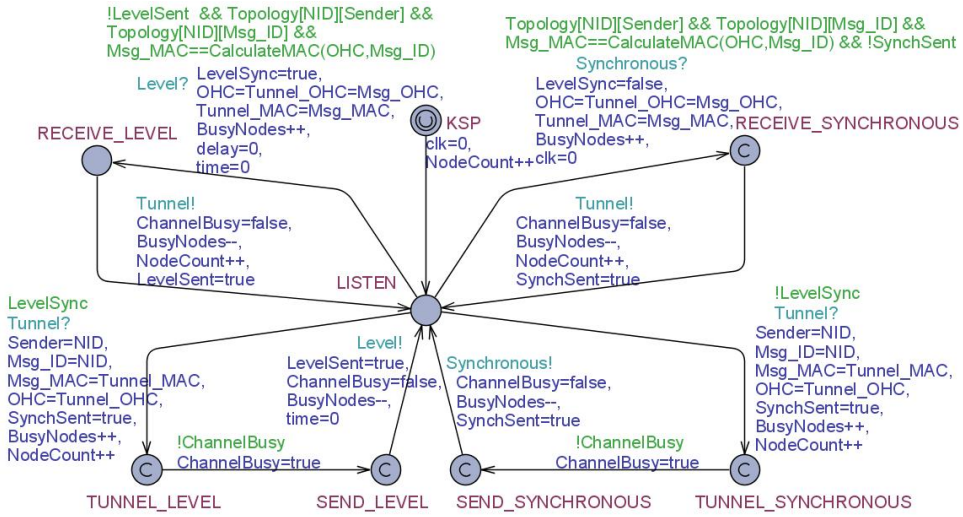


Figure 6.14: Attacker model to check the tunnel attack

are propagated very fast (retransmitted within 10 ms) the LEVEL beacons are a potential prey to the rushing attack launchers. The LEVEL beacons propagated in advance will enable nodes to assign incorrect levels to itself and to its neighbours. This will enable longer routes than normal and thus the average hop count the data travels will increase.

The nodes perform bidirectional verification in the KSP. Thereafter only messages from the verified neighbours are accepted. Any attempt to relay data far away without a wormhole or tunnel will be avoided straight away. The issues related to hello flood attack, that can enable rushing attack, have already been addressed in Section 6.2. The INA and wormhole attack have been avoided (Section 6.3) before any of these messages are sent. The solution for a tunnel attack has also been incorporated as discussed in Section 6.5. Thus all possible doors for the rushing attack have been closed in RAEED. The formal model presented in Section 5.6.9, which was later extended in Section 6.4.1 and 6.5.1, can be used to verify that the rushing attack is unsuccessful in RAEED. The proof of the property in Equation B.1 confirms the resistance to the rushing attack.

6.7 Prevention Against the Black hole Attack

It has been proved using the formal framework (Chapter4) that both INSENS and LEAP are susceptible to the black hole attack in the Data Forwarding Phase. It was also confirmed that a black hole is possible in Arrive both by employing a single path (Section 4.9.3.1) or with multiple paths (Section 4.9.3.2). Note that the black hole is not possible simply because of an INA and wormhole, which have been addressed by the current thesis earlier, but also because

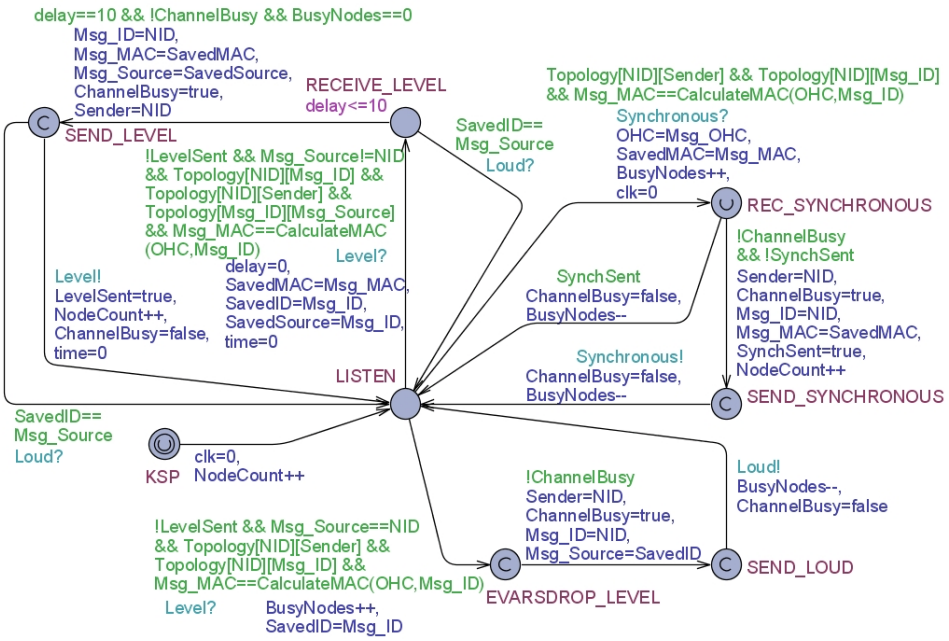


Figure 6.15: Modified node model to check the tunnel attack

of node capture. A node can be captured at any stage which may then act as a black hole attacker.

This section confirms that the black hole can be pacified using the neighbourhood watch approach adopted by RAEED. Each node observes the performance of its neighbour after forwarding data to that node. This eavesdropping enables nodes to detect if the neighbour has forwarded the data further to a legitimate 2-hop node. Each node builds a log of data sent (forwarded) and received by eavesdropping (further forwarded to another node) for each individual neighbour. The neighbours are then ranked based on these points. When forwarding data, a neighbour with the best ranking is chosen as a potential data forwarding node. In the case of equal ranking, the node's level (hop distance from BS) is considered. When nodes have equal rank and level, then any neighbour among them is randomly chosen. A better option here is to select a node that has been sent the least number of messages to provide load balancing in the protocol. This ranking system is the subject of further improvement in future work.

The thesis proves that the employed neighbourhood watch scheme can solve most of the black hole issues. The formal model confirms that the black hole possibility in ARRIVE has been removed. The lost mechanism employed by the scheme informs the higher level node that there is no further node available for forwarding the data and thus the lost beacon enables the higher level node to forward data again to another neighbour. The formal modelling and simulation results also confirm that RAEED has higher throughput than INSENS in the presence of a black hole.

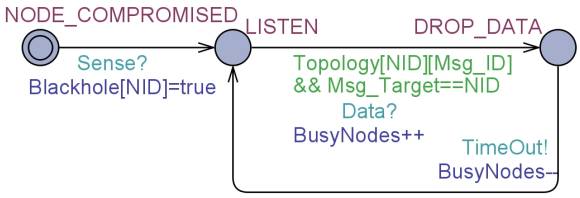


Figure 6.16: Black hole Attacker model

6.7.1 Formal Verification

The hypothesis for this model is that:

"The Black hole Attack is unsuccessful in RAEED".

The assumptions used to evaluate the DFP in Section 5.7.2.1 are retained here. However, for networks greater than 5 nodes only the grid topologies and topologies that were proven to be successful for black hole attacks in previous protocols in Chapter 4 were checked.

The **model** remains unchanged for ***node*** and ***sink*** as was used in Section 5.7.2.1. An ***attacker model***, however, is introduced for the black hole attack and is shown in Figure 6.16. The attacker model starts in the NODE_COMPROMISED location indicating that node has been compromised and thus the `Blackhole` flag is set. This is a simple black hole model that drops data upon receiving it. So the model just moves between the two locations the LISTEN and the DROP_DATA when the data is received. Note that a timeout is triggered by the attacker which indicates that the attacker has not forwarded the data for a specific time threshold. As before, this was done to save state space.

The **verification** involves proving the same properties as described in Section 5.7.2.1. The successful proof of all properties confirms that the black hole attacker has no effect on RAEED and the throughput remains 100%.

6.7.2 Simulation Results

6.7.2.1 A 1000 Node Network with Black holes

It has been discussed in Section that INSENS is vulnerable to the black hole attack in spite of there being 4-BS and 4 paths. The results were worst when low density networks were employed. To confirm that the low level INSENS model was correctly developed, the results obtained were compared with published INSENS results [19] in Figure 6.17. As the research is limited to a network of 1000 nodes and an average density of 16, employed in published INSENS results, cannot be achieved using grid network (Section A.2.1.3), the research employed 1000 node

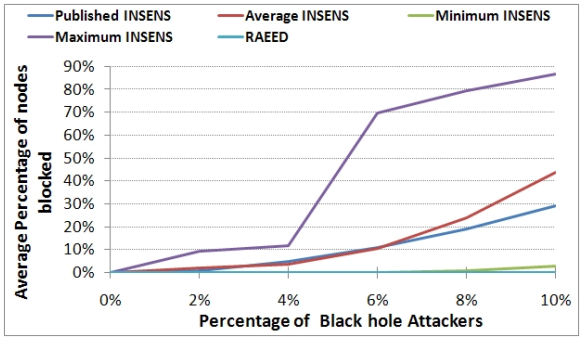
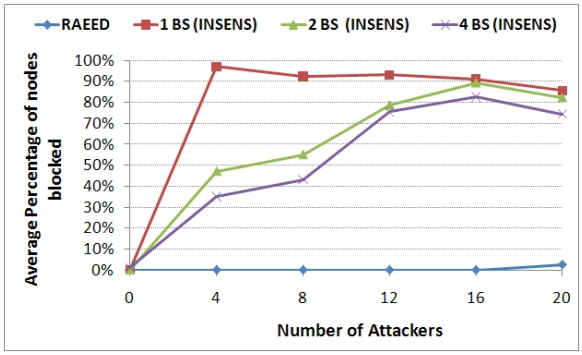


Figure 6.17: Percentage of nodes blocked due to black hole in 1000 node network

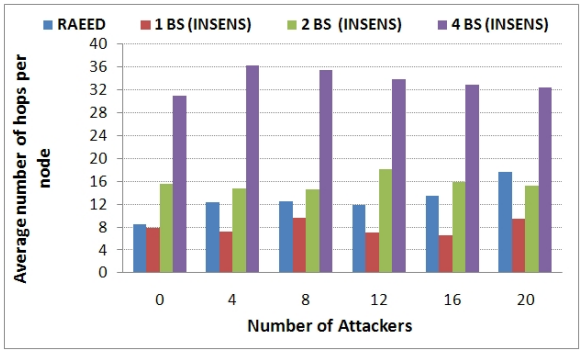
network with an average density of 20. The experiments were repeated 30 times and results were then plotted for maximum, minimum and average values as appears in Figure 6.17. For comparison the results plotted in INSENS [19] and the results achieved by RAEED under the same conditions are also plotted. Note that the x-axis is the percentage of attackers deployed which was what effectively plotted in INSENS [19] for 2000 nodes. The percentage of legitimate nodes blocked is the percentage of nodes whose data was not received by BSs.

It is evident that the average results obtained here for the INSENS protocol are almost the same as those presented in INSENS [19] for a lower percentage of attackers. The results vary for a higher percentage of attackers. The reason for the variance is because the research already proved that the position of attacker is very critical and determines the number of data packets lost. INSENS authors do not mention this effect in their protocol. As the number of attackers is increased the variation between the results also get increased. A number of experiments were performed (for fewer numbers of attackers), to vary attacker position and thus got average results similar to what had been published. However, as the number of attackers was increased these experiments became more difficult to conduct (in a 1000 nodes network there are 100 attackers for 10% attackers). Finally, the maximum value attained in the experiments was always more than the published average value and minimum value attained was always less than the published average value. This confirmed that the INSENS protocol implemented in nesC can replicate previously published behaviour.

Considering Figure 6.17, it is obvious that in spite of an increase in the percentage of compromised nodes (black holes), the percentage of blocked nodes is 0% for any percentage of attackers. Moreover, the protocol employs a single path for data routing instead of 4 paths and 4 BS used by INSENS. This indicates that RAEED is more robust and possess lower overheads than INSENS.



(a) Percentage of nodes blocked due to black hole



(b) Average number of hops taken to reach BS

Figure 6.18: Case 1:The effect of the black hole attack on INSENS and RAEED in a 200 node network with a density of 8 and asymmetric attacker position

6.7.2.2 A 200 Node Network with Black holes

These experiments tested INSENS and RAEED in a 200 node network with a non-boundary density of 8. The experiments were repeated 30 times for both INSENS (1 BS, 2 BS and 4 BS) and RAEED. Two graphs are plotted after experiments; the first one shows the percentage of nodes blocked in the presence of attackers and the second plots the average number of hops the data messages travel before reaching the BS. The second parameter is effectively the message overhead involved in routing.

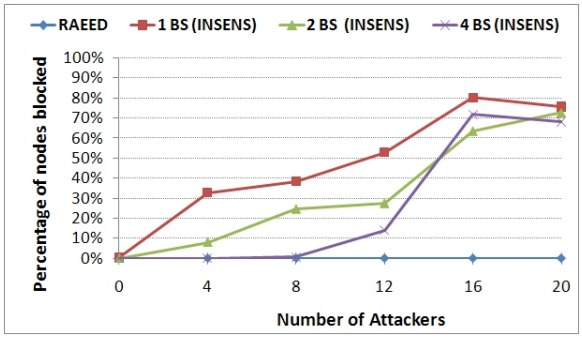
The experiments were performed in 2 different ways. In ***Case 1*** the attacker's position is not always the same for all BS. Note that as BSs are placed at the four corners for a 4 BS network, the attackers must be distributed at each of the four corners, because if the attackers are not present near all BSs, any one of the BS receiving a data message will enable 100% throughput. For a fair test each BS must have an attacker near it. In the first set of experiments the attackers are only placed near BSs. So for 2 BS networks half of the attacker nodes are the same as those of 4 BS but other half are different as there is no need to place attackers near the other 2 corners containing no BS. Thus the rest of the attackers are concentrated near the 2 BS. Similarly for a single BS the attackers are only concentrated near the single BS corner. This means that only half of the total attacker nodes are similar as in the case of 2 BS and a quarter of the total attackers are same as were on 4 BS network. The attacker position for RAEED and 1 BS INSENS protocol were the same as both contain a single BS. The attackers are first placed at a 1-hop distance from BS, then at a 2-hop distance and so on. In all the cases it was always ensured that at least one legitimate path to each BS was available. The attackers are placed in symmetric positions in case of multiple BSs.

The results for ***Case 1*** are shown in Figure 6.18(a) and Figure 6.18(b). Figure 6.18(a)

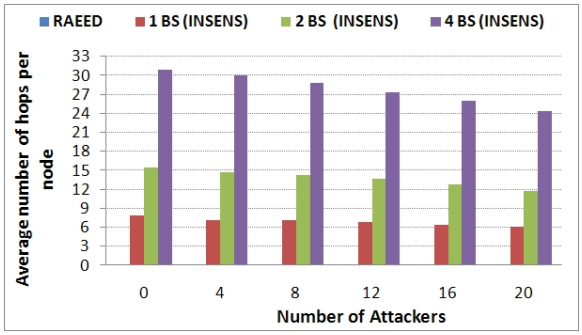
shows that the average number of nodes blocked increased with the increase in percentage of attackers for all INSENS cases. However, for the case of RAEED no node is blocked up until 8% attackers and only 2.5% of legitimate nodes are blocked when the number of attackers is 20. Moreover, the throughput is high (blocked nodes are fewer) in a case where more BSs are employed in the INSENS protocol.

Figure 6.18(b) displays the average number of hops that the data messages took before reaching the BS. This effectively is the total number of hops taken divided by the total number of nodes (200). As anticipated, the value is doubled in the INSENS protocol each time the BSs are doubled. This is because the paths are doubled and thus data is sent multiple times. Note however that the average number of hops decreases for all networks, in the INSENS protocol, as the percentage of attackers is increased. This is again expected since an increased number of attackers cause more messages to be lost before reaching their destination decreasing the average hop count. Conversely, the average number of hop counts for RAEED increases with the percentage of attackers. Comparing it with the INSENS results, the average hop count is increased from 12 to 16; whereas the average hop count for INSENS for 1 BS and 2 BS is 12 and 22 respectively, in the absence of any attacker. This means that the actual increase in message overhead is 33% in the presence of 10% attackers for RAEED. The reason for this increase, in the presence of attackers, is that some messages are lost and thus nodes try to resend data starting from the point of loss. In the presence of many attackers this attempt may be made many times increasing the average throughput. This was not possible even in the presence of 4-BS which had 4 times the message overhead. Finally, it is noteworthy that this message overhead occurs only during the first phase. In the next phase most of these black holes had been detected and thus were avoided by the nodes. It enabled a lower overhead and an increase in throughput.

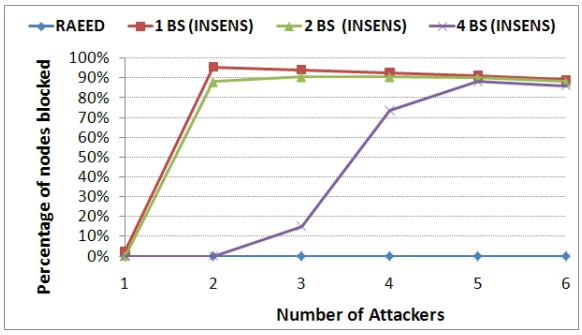
For ***Case 2***, as explained earlier, the position of attackers remains the same for all the experiments whether it uses a single BS or multiple BSs. The results are shown in Figure 6.19(a) and Figure 6.19(b) respectively. The average hop count result pattern remains the same with the increase in number of attackers. However, the average number of nodes blocked (Figure 6.19(a)) are different from those obtained using the INSENS protocol depicted in Figure 6.18(a). This emphasises the importance of attacker position in the network. The attackers are distributed at each of the four corners, whereas in Case 1, the attackers were concentrated near the available BSs. So, even for 1-BS and 2-BS networks, the results improved for the lower percentage of attackers. The throughput for RAEED, however, gets even better. No legitimate node is blocked and the throughput of 100% is achieved. The maximum and minimum throughput, for INSENS and RAEED, is also plotted in Figure 6.19(c) and Figure 6.19(d) respectively. These results confirm lots of variations for the INSENS protocol. However, the results for RAEED remain stable at 0% nodes blocked.



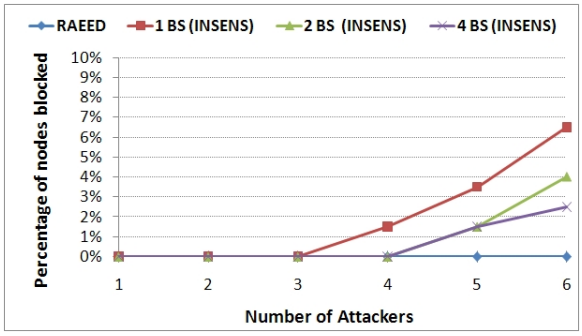
(a) Percentage of nodes blocked due to blackhole



(b) Average number of hops taken to reach BS



(c) Percentage of nodes blocked due to blackhole



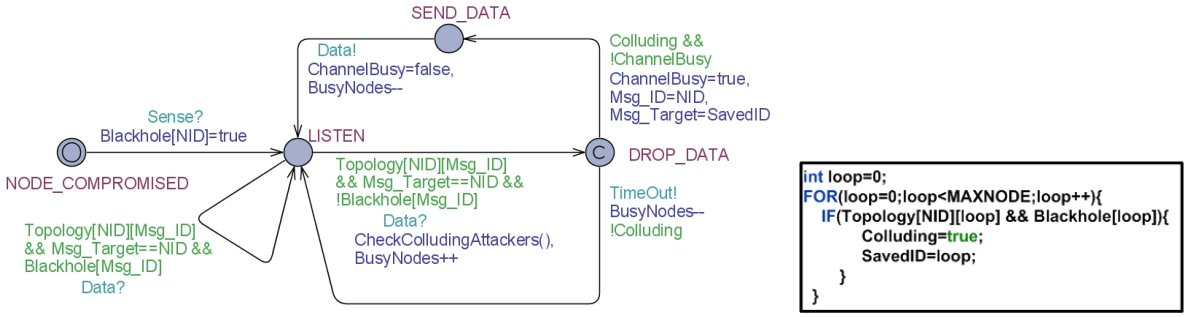
(d) Average number of hops taken to reach BS

Figure 6.19: Case 2: The effect of the black hole attack on INSENS and RAEED in a 200 node network with a density of 8 and symmetric attacker position

6.7.3 Intelligent Black hole Attack Prevention

In this section, the black hole attacker is made more intelligent to check its effect on RAEED. The topologies containing multiple black hole nodes where ARRIVE has failed were checked. It was confirmed that RAEED can successfully prevent the black hole attack in these topologies. The protocol thwarts the attack by employing a single path instead of multiple paths used in INSENS and ARRIVE to prevent the black hole. This gives RAEED an edge over the previous protocols. A black hole attacker with a spoofing attack will also fail here if an intelligent black hole attacker sends data to oblivion i.e. to a non-existent node because the protocol will consider a node to be a black hole if the data is not sent to a valid node. Note that 2-hop neighbours were recorded in the NPP by exchanging NEIGHBOR messages and such an attacker will be spotted straight away.

This, however, does not prove that the black hole is not possible in RAEED. The formal framework detects a flaw in RAEED. An intelligent black hole attacker, if it has another attacker node as a neighbour (which is not within the range of a sender), can always forward data to it



(a) Attacker Model

(b) CheckColludingAttackers() Function

Figure 6.20: Intelligent black hole attacker model

and the colluding attacker can drop the data packets. The legitimate nodes observe the first black hole node acting in correct manner and will never lower its ranking. The second attacker's cannot be observed by the sender of data because it is two hops away. In this case the black hole will be successful. A modified ***attacker model*** is shown in Figure 6.20(a). The attacker, when it receives a data packet, checks if there is another attacker present within its range using function `CheckColludingAttackers()`. The decision to drop data is then made depending on if the colluding attacker is present or not (`Colluding` flag). If an attacker is present within range, the data is always sent to that node. The attacker node will only drop data coming from another attacker node. There are a number of solutions to this problem:

One solution is by providing multiple paths to avoid this kind of attacker. However, a modified node model which uses multiple paths confirms that there is a case where all data packets still go through the colluding nodes if they are near the BS. The legitimate nodes still remain unaware of the situation throughout. Thus braided multiple paths (some nodes share multiple paths) is not a general solution to this kind of attack. If the paths were disjointed, however, then the colluding nodes would have been avoided. But multi-paths will always add extra overhead which our research attempts to avoid.

Another solution to this intelligent attack is to employ LOUD acknowledgements from 2-hop nodes. This incurs additional traffic overhead however. Another solution is to send a FAIL message when any neighbour has not forwarded its data. That will inform the 2-hop distant nodes that the data has not been forwarded and that the sender node should regenerate the data. This will reduce data traffic and messages would be broadcast only when a successful attack is detected. However, this could lead to framing attack and also violates one of the

assumptions made that the data forwarding decisions are taken locally and independently by each node. The solution to framing is to act only if most of the 2-hop neighbours broadcast this FAIL message excluding the node to which the message has been forwarded. All nodes have information identifying which 2-hop nodes are neighbours of the target node after NPP. Thus, after performing this check, the legitimate nodes will lower the target node's ranking. The sender node then regenerates data to another neighbour. The legitimate node only lowers the ranking of the suspected node and does not totally avoid it afterwards (by declaring it as an attacker). This is because the message could have been lost due to noise or collisions as well. Moreover, by only lowering the ranking, the impact of framing attack will be reduced. The only unresolved problem in this solution is the RF fading effect as the time passed and traffic overhead, specially when the data is sent frequently.

A solution to above problems is to broadcast a FAIL beacon to 1-hop neighbours using the normal power transmission. In this case some of the nodes transmitting FAIL will be immediate neighbours of the sender. This will enable the sender to retransmit data again to the same target node and observe if the target still transmits data to the suspect node in the future. Note that if the target is not the black hole attacker; it should avoid the suspected node next time because the ranking of the suspected node would have been reduced. Thus by keeping the target under observation for some time, this colluding attack can be detected and the attacker can be totally avoided. There are two further advantages in this scheme. First the framing attack is not now possible, because neither the ranking is lowered nor the node is added as a suspected neighbour. On the contrary, the neighbour is kept under observation for some time and is deleted from the NT on confirmation that it is acting maliciously. Secondly, as FAIL is broadcast, the neighbour nodes will also observe the suspected node immediately rather than waiting until that suspected node acts maliciously on their data. Thus the solution can quickly detect the colluding black hole acting intelligently and can fail this attack. The message overhead is also low for this scheme. However, it will fail in case the node density is so low that the colluding nodes have no common legitimate neighbour. In that case no legitimate node can detect that the data has been dropped and thus FAIL cannot be generated. This setback exists with the earlier 2-hop FAIL broadcast as well.

The solution to all the drawbacks in previous schemes is to use load distribution in data forwarding. A node must not forward all data packets to the same neighbour and a node performing that act must be avoided. This will give two advantages. First the life of the network will increase as the load is distributed in all nodes. Second, the colluding black hole will be less successful. However, if the attacker is aware of this situation it can act smartly and perform the node distribution as the other nodes do. In that case the black hole is not fully avoided but only pacified and some packets might be lost through that colluding black hole without detection. Also the load distribution will enable some routes to take a longer time than normal ones. This will add latency and will engage more nodes than in the absence of

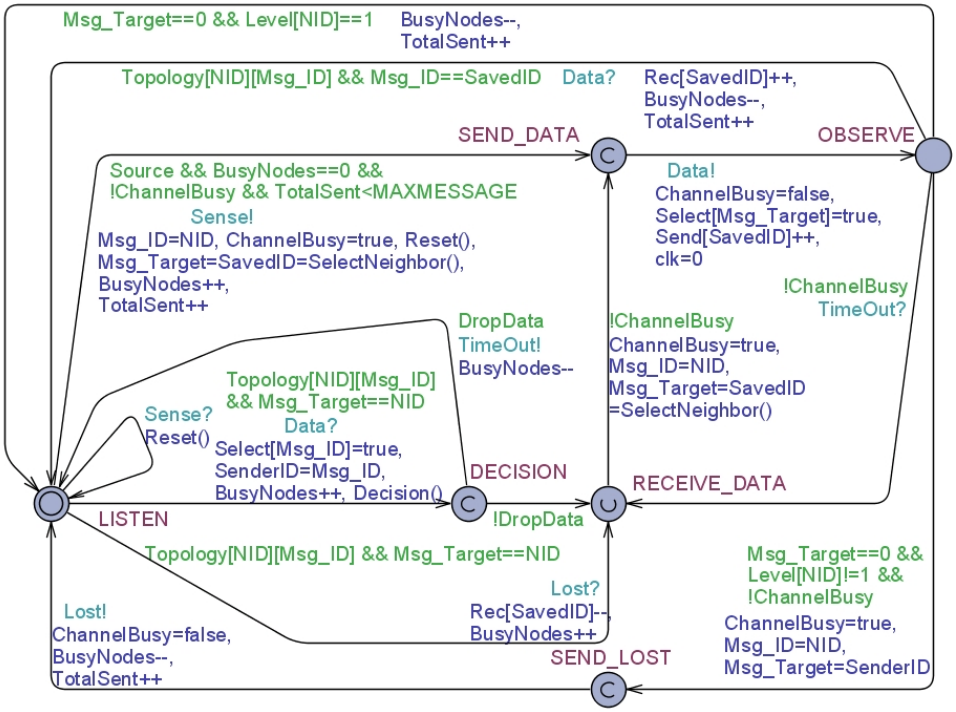


Figure 6.21: Gray hole attacker model

load balancing. Most of these modifications discussed are left as future work.

6.8 Prevention Against the Gray hole Attack

6.8.1 Formal Verification

The hypothesis for this model is that:

"The gray hole or selective forwarding attack is unsuccessful in RAEED".

The assumptions and ***models*** remain the same for node and sink models as those for the black hole attack prevention in Section 6.9.1. Only the attacker model is modified as shown in Figure 6.21. The attacker model is a combination of the node model and the attacker model (black hole). Since the gray hole attacker selectively drops data packets and acts normally otherwise. This decision is made by the attacker by keeping a history of the data it has received (`TotalRec`) and forwarded (`TotalSent`). The model includes a threshold percentage named `PERCENTAGE`, which determines what percent of data messages will be dropped. This option is adopted before the DECISION location. The attacker then either drops data or forwards it further by behaving like a normal node.

The **verification** involves proving the same properties as described in Section 5.7.2.1 and Section 6.9.1. The successful proof of all properties confirms that the gray hole attacker has no effect on RAEED and the throughput remains 100%.

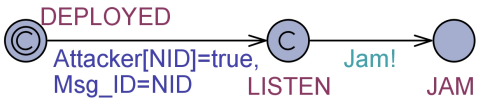


Figure 6.22: Attacker model to test the Jamming Attack

6.8.2 Simulation Results

The computer simulation were performed to confirm the effect of selective forwarding on RAEED. As expected, the results were similar to those observed for the black hole attack. The experiments performed were a scalability check (1000 nodes) and the effect of density on a 200 node network, as done for the black hole attacker in Sections 6.7.2.1 and 6.7.2.2. The results were checked for different percentages of attackers up to 10%. The attackers transmit 50% of data packets and block the rest. It was observed that the gray hole attacker failed to create any impact, the throughput remaining unchanged. The average hop-count, however, was reduced because the attackers forward only half the packets correctly. So a smaller number of data packets were resent thus reducing the average hops it took before reaching the BS.

6.9 Prevention Against the Jamming Attack

6.9.1 Formal Verification

The hypothesis for this model is that:

"The Jamming Attack is unsuccessful in RAEED".

Apart from the assumptions stated in Section 4.3.3, it is further assumed that nodes have already successfully completed KSP and RSP. Instead of testing all possible 5 node topologies, the grid topologies of 16, 25 and 36 nodes were checked. The reason for this strategy is that the jammer will jam neighbour nodes so small networks cannot guarantee a legitimate path in presence of an attack.

The **sink model** used is the same as was for the black hole attack (Section 6.9.1). The new ***attacker model*** is shown in Figure 6.22. It has two committed states and thus generates a `Jam` message before moving to final state (JAM). This `Jam` signal jams all the neighbours of the attacker. The ***node model*** used earlier in black hole attack is modified to accommodate the ACK message as explained in Section 5.7.3, shown in Figure 6.23. The additional two locations are used to show that the node is jammed. Note that the `Timeout` message is used to indicate no action is taken on any data, thus the jammed node generates that message upon receiving

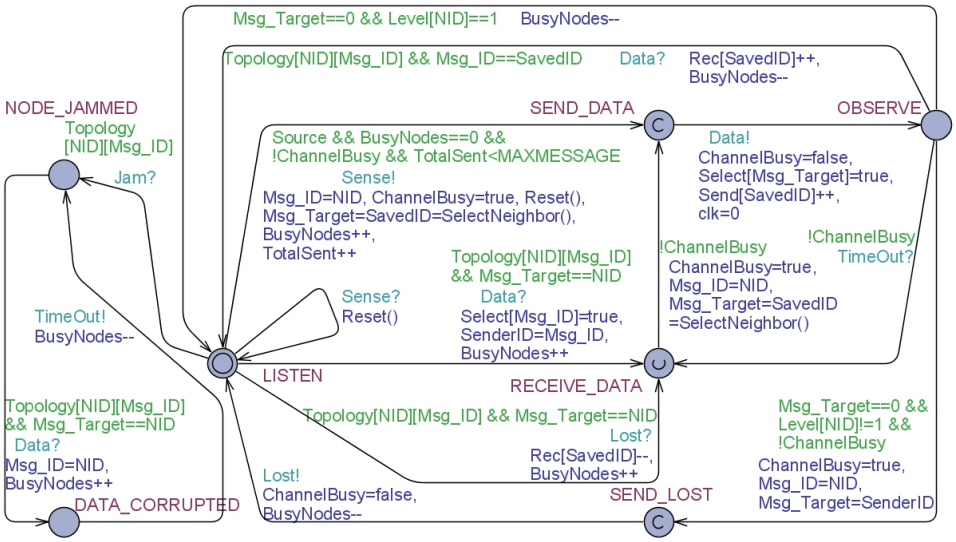


Figure 6.23: Node model to test the Jamming Attack

any data in the jammed state.

The **verification** process involves the same properties as in Section 5.7.2.1. All properties were proved thus confirming that the jamming attack remains unsuccessful in RAEED. The data transport property confirms that the messages received at the BS are the same as those generated by the source node.

6.9.2 Computer Simulation

Computer simulation were performed on 200 and 1000 nodes grid networks. The density was varied for each case between 4 to 28 non-boundary nodes. Two jammer nodes were placed near the opposite corners of the network, one jammer near to the BS and the other at the opposite corner. The jammer's range was limited to 1-hop i.e. a jammer can block all its surrounding (radio range) nodes. It was observed that apart from 4 node density networks, the average number of blocked nodes remained 0%. The data was routed through the other nodes once it was lost in the jammed area. Even the jammed nodes were able to send data to the BS, because the jammed nodes broadcast data one by one to each neighbour and it eventually was received by a node outside the jammer's range. The only problem was that the jammed nodes did not receive the feedback and thus will kept on sending the data to all neighbours. This drawback can be removed by limiting the number of attempts a node makes for forwarding any data. The consequences, however, would be that the jammed node's data will not reach the BS. This is left as a future decision regarding the importance of jammed area data or energy savings. For the 4 density networks, the average number of blocked nodes are increased to 5%. This is because of the smaller number of the available neighbours.

6.10 Chapter Summary

This chapter rigorously evaluates RAEED against different DoS attacks as was done in Chapter 4 for the other routing protocols. The bidirectional verification solved the *hello flood attack* and computer simulation as well as hardware implementation confirmed that not a single node is affected by the hello flood attacker. Only the legitimate nodes were tagged as verified neighbours. A unique and innovative scheme is presented to solve both the *INA* and the *wormhole attack*, which utilize the transmission power to communicate with 2-hop neighbours and detect any virtual link between them. The results were verified using formal modelling and hardware implementation. The formal frame work also detected some vulnerability of new schemes against an intelligent wormhole attacker, which utilizes cryptographic knowledge to fail the scheme. Although such a wormhole/INA attacker has not been discussed before, the chapter provides a solution for this probable attacker. It has been proved formally that the attacker will be unsuccessful even by using the cryptographic information. The solution of wormhole/INA is a unique contribution to secure WSN research.

The sinkhole and tunnel attack solutions have also been verified using formal modelling and computer simulation. These attacks were proved unsuccessful in RAEED. Furthermore, a solution for a tunnel attack, in combination with framing attack, has also been incorporated in RAEED. The rushing attack is automatically solved once hello flood, wormhole, INA and tunnel attack have been resolved. The proof of a property that nodes always get their correct levels proves that the rushing attack has been unsuccessful. Finally, the neighbourhood watch scheme in DFP is evaluated exhaustively against black hole, gray hole and jamming attacks using the formal modelling. The attacks are later analysed on different size networks for varying densities, using computer simulation. The number of nodes blocked as a result of these attacks were compared with the INSENS protocol. It was confirmed that RAEED, in spite of its low message overhead (single path), present few nodes to be blocked as compared to INSENS with multiple paths and BSs. Some intelligent black hole attackers have also been presented, which had been detected by the formal framework; these can fail the new scheme. The solution for those kind of attack has also been annotated and proved using formal modelling.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This research work applies formal model checking to investigate the security of various wireless sensor network (WSN) routing protocols. This research suggested that the formal modelling approach is a useful verification process and should be performed at an early development stage (design phase) so that any hidden error present in the design can be rectified and removed. This research, therefore, first formally defined most of the published and widely recognized DoS WSN attacks, a task which had not been done before. Researchers have used different definitions for the same attacks and no concise/formal definitions have been available to them. This research has filled this gap by writing the specifications of DoS attacks in a formal way. Once the specifications of the attacks had been outlined (using Z specifications), it significantly has assisted in developing a formal framework, which has been used for checking various routing protocols. A formal framework was then implemented to perform rigorous testing of recognised routing protocols against the DoS attacks. The same criteria has been adopted for evaluating the newly developed protocol.

In summary, the most potent available technology has been used to verify various routing protocols using a formal framework. The framework can automatically generate traces in the cases where an attack is successful in any possible topology of N nodes. Some of these protocol failures against the DoS attacks have already been reported in the literature [15, 234]. However, the current modelling method has successfully detected these failures automatically. This has the potential to save a lot of calculation time for the future detection of attacks through visual inspection or computer simulation. Moreover, some worst cases have also been detected, which to our best knowledge, have not been reported. Among these sinkhole attack which prevents data from reaching the BS in TinyOS, a bug in Rumor Routing and a worst case in the Direct Diffusion. Secure protocols have also been analysed such as Authentic TinyOS Beaconing (using uTesla), LEAP/INSENS, Arrive and ARAN. Although the vulnerabilities of Authentic TinyOS have already been identified earlier by researchers, some undiscovered faults in INSENS, Arrive

and ARAN protocols have been exposed, such as their vulnerabilities to the black hole/gray hole attack, INA and wormhole attack. The results obtained in the formal framework have improved the confidence that the framework can successfully model and verify if an attack is possible in a routing protocol even though such models have been limited to just 5 nodes. One may use the proposed framework to analyse new or existing protocols. The framework was later enhanced to test up to 36 nodes. Finally, the framework has been verified by implementing the successful attacked topologies of INSENS in the TOSSIM to validate the previous results.

Arrive protocol provides robustness against malicious and failed nodes by maintaining a neighbour reputation. The developers of the protocol themselves suspected that there would be an abundance of additional messages when using passive participation due to the hidden terminal problems. Also, unidirectional links were another cause of concern. Finally, the Arrive protocol assigns levels to each node based on their hop distance from the BS, their developers were aware that malicious nodes may lie about their level or replicate these levels to draw more traffic (spoofing). In addition, the attackers may forward the packets into oblivion to increase their reputation (sinkhole). Despite these problems, the formal framework has confirmed that Arrive is still vulnerable to the black hole attack when a node in the network has only black hole(s) as parent(s) and neighbour(s). A black hole is also not detected by off-spring nodes when they have forwarded the data, leading to all the down stream traffic being lost. The formal framework has also confirmed that the wormhole attack will assign incorrect levels to the nodes enabling the data to be sent to oblivion. Because a wormhole is usually composed of multi-hop tunnels, the passive forwarding will also not be possible since a tunnelled node will not lie within the radio range of these nodes. In addition, a hello flood attack is also possible in Arrive. However, the problem of INA has been addressed successfully using passive forwarding, as confirmed by our formal model. The developers of Arrive were unaware of the fact that passive forwarding has solved INA. This research has demonstrated that the developed framework can detect this previously unknown attack. Finally, the formal framework has also confirmed that, as indicated by its developers, the Arrive protocol is susceptible to both sinkhole and spoofing attacks.

A robust routing protocol ARAN, which employs the public key cryptography, has also been evaluated using the proposed formal framework. ARAN does not utilize the hop count and accepts the fastest links to avoid attacks. The formal framework has confirmed that there are some weaknesses in ARAN. Cryptographic techniques alone cannot defeat the INA and the wormhole attack. By creating virtual links the routes can be corrupted. The proposed formal model is able to automatically detect this type of attack in ARAN and generate the trace indicating why and how ARAN fails in the presence of INA/wormhole. Moreover, the nodes can be captured and a compromised node may act as a black hole attacker. The formal model also confirms the vulnerabilities of ARAN to the black hole attack. The proposed model confirms that, in spite of the data-signature mechanism, the data may not reach the target

nodes.

The developers of INSENS claim that the Enhanced INSENS has solved vulnerabilities to many attacks by using authentic broadcast and multiple paths for data propagation. Formal modelling and later the simulation results refute this claim. It has been successfully demonstrated that INSENS is vulnerable to many DoS attacks such as the black hole, INA, wormhole etc even in the presence of an ideal channel (with minimum collision and no noise), multiple paths and a small network. A rushing attack can easily be launched during the request propagation period, once a wormhole is successfully in place. This proves that the findings of [19] are not correct. However, a bidirectional verification has successfully removed the hello flood attack. It was later confirmed that the black hole attack can cause a low throughput especially in low density networks. Even in denser networks than those adopted in [19] and with a lower percentage of attackers, many messages can be blocked during the data forwarding phase, especially if the attackers are near each sink, even under ideal conditions. Further experiments have confirmed that the throughput suffers in large networks with dense deployment. The effect of attacks will be more detrimental if noise and other environmental effects are also considered.

Chapter 4 has shown that by using formal modelling, the hidden bugs in routing protocols can be detected automatically. Next, a new routing protocol, RAEED, has been developed; this has been rigorously shown to be resistant to DoS attacks. It was designed to mitigate the effects of the most powerful DoS attacks such as the black hole, the hello flood, the gray hole, the INA, the jamming, the rushing, and the wormhole attacks. RAEED can also resist against the sinkhole and the tunnel attacks in the presence of strong encryption. It is worth noting that the research was not intended to develop any encryption mechanism, rather it assumed that an encryption technique is already in place. The protocol design has been rigorously tested using formal modelling to remove the hidden bugs/errors. The computer simulation were later performed to support and quantify the results as well as to check the effects of scalability and node density. Moreover, different phases of RAEED were tested in the presence of noise using formal modelling and simulation. The complete protocol is divided into 3 main phases: KSP, RSP and DFP. The KSP involves the exchange of keys and bidirectional verification. As the protocol assumes that an encryption mechanism already exists, the KSP of INSENS and LEAP protocols are adopted as the foundation. It was also shown that the KSP presented in RAEED is more efficient as it provides the same security properties and has a lower message overhead. The keys were piggy bagged in the bidirectional verification phase and a three way message exchange performed to reduce data traffic. The RSP involved an authentic level assignment to each node using the hop distance from the BS. The nodes also perform neighbour ID exchange and the loud test to verify 2-hop neighbours in order to remove any virtual connections between the nodes. The authenticity was provided using a one way hash chain known only to the BS. Each sub-phase was tested separately, through extensive use of the formal modelling and simulation for larger networks. The DFP involves the propagation of the data from the source to the sink.

Node reputation and the level was used in data routing. The nodes were ranked based on their performance by eavesdropping their activities after forwarding the data. A lost message was generated if a legitimate node could not forward the data further. This enables the predecessor node to forward the data to another path. The scheme was evaluated using formal modelling and computer simulation. Finally, the effect of noise on RAEED was evaluated and it was confirmed that the protocol can work in the presence of both multiple paths and BSs. However, a single BS and path are used to reduce the number of messages. It was proved that the protocol is robust, works well in noisy conditions as well as in different sizes of networks with varying density.

Finally, RAEED was rigorously evaluated against different DoS attacks by employing the formal framework with the support of either the computer simulation or a hardware implementation to demonstrate the findings. The DoS attacks verified included the hello flood attack, the wormhole attack, the INA, the sinkhole attack, the tunnel attack, the rushing attack, the black hole attack, the gray hole attack and the jamming attack. The formal modelling confirmed that the bidirectional verification has solved the hello flood attack. The computer simulation and the hardware implementation have concluded that not a single node has been effected by the hello flood attacker. Only legitimate nodes were tagged as verified neighbours. A unique and innovative scheme has been used to solve the INA and wormhole attack, utilizing transmission power control in communicating with 2-hop neighbours to detect any virtual link. The results were verified using both formal modelling and hardware implementation. The formal frame work later detected some vulnerability of new schemes against an intelligent wormhole attacker, which utilizes cryptographic knowledge to fail the scheme. Although such a wormhole/INA attacker had not been identified before, this thesis proposes a solution for this type of attacker as well. The sinkhole and the tunnel attacks, both simple and complex, solutions have also been proved unsuccessful using the formal modelling, when RAEED is used. The rushing attack is automatically solved once the hello flood, wormhole, INA and tunnel attack have been pacified. Finally, the neighbourhood watch scheme in the DFP was exhaustively checked against the black hole, gray hole and jamming attacks using the formal modelling. These attacks were then considered on different size networks, for varying densities and scalability, by experimenting on the computer simulation. The number of nodes blocked because of these attacks were compared with the INSENS protocol. It was confirmed that RAEED, in spite of low message overhead (single path), enabled fewer nodes to be blocked as compared to INSENS with the multiple paths and BSs. The intelligent black hole models were also considered and the formal framework was used to confirm that they would remain unsuccessful in RAEED.

7.2 Summary of the Contribution to Research

7.2.1 Main Contribution to Research

The principal research contributions of this work are:

- The development of a new routing protocol [235] that works better in the presence of the attacks including hello flood, rushing, wormhole, black hole, gray hole, sink hole, INA and jamming.

- The formal Specification/definition of the published WSN attacks. These specifications present an abstract model for different attacks. More detailed models can be developed later which, in conjunction with the specifications of a routing protocol, can help researchers in detecting the vulnerabilities of those protocols against different DoS attacks. These definitions are also more widely applicable to ad-hoc networks and MANETs. These results are expected to be published in [236].

- A unique and innovative defence against the wormhole attack and the INA has been presented. It does not require any additional hardware and it is well suited to WSNs. Adaptive power transmission is used to confirm virtual links between the nodes. It has been proved formally that the attacker will be unsuccessful even if they use cryptographic information to launch these attacks. This solution can be incorporated in any protocol and, is expected to be published in [237].

- A formal framework [238, 239, 240, 241, 242] has been developed to check the vulnerability of different routing protocols. The initial framework was based on Andel's framework [37]. However, a much improved framework was later developed to accommodate many other routing protocols within the limitation of state space explosion. The formal framework has detected the worst cases for different routing protocols which to our knowledge have not previously been detected.

- A formal verification of routing protocols that had not been formally studied previously against the DoS attacks, has been performed. These include TinyOS Beaconing, Authentic TinyOS using uTesla, Rumour Routing, LEACH, Direct Diffusion, INSENS, ARRIVE and ARAN. These results are published in [238, 239, 240, 241].

- Vulnerability has been detected in the widely regarded secure/robust WSN protocols INSENS [241], Arrive [239] and ARAN [240].

- An innovative design analysis of using a combination of formal modelling and the simulation has been presented [235] to evaluate the robustness of existing and the new protocols against all possible attacks.

7.2.2 Other Contributions to Research

- A neighbourhood watch scheme has been presented to solve the black hole [242] and the jamming attacks.
- An improved KSP is presented that has low message overhead compared to INSENS.
- A worse case has been discovered in both Direct Diffusion and Rumour Routing. These result are published in [238]
- The protocol presented can not only be used as a solution against the DoS attacks, it is equally applicable for WSN applications placed in the noisy environments as the protocol is robust against noise.
- The solution presented in this thesis to solve DoS attacks and especially the black hole attack, also automatically solves misbehavior and selfish node problems. It also isolates the dead and failed nodes. The ranking of the dead nodes will drop and thus they will not take part in the future routing of data. Therefore, the new protocol, RAEED, offers an excellent solution for insecure WSNs as well.
- This protocol is also a solution for the neighbour nodes whose transmission range deteriorates with time. Their ranking will be lowered immediately when the throughput starts to deteriorate. The neighbours whose links become unidirectional are also removed automatically as nodes no longer select these nodes for data forwarding.

7.3 Future Work

The formal specifications of DoS attacks written in Z present an abstract model for different attacks and can be a starting point for future improvements for new attacks on wireless networks. Further detailed models may be extracted in conjunction with the specifications of a routing protocol to assist researchers in detecting the vulnerabilities of the protocols against different DoS attacks.

Encryption issues have not been addressed by this research. The KSP involves the exchange of keys between nodes. Different encryption mechanisms, both symmetric and public cryptography, may be examined in future. DoS attacks such as the sinkhole attack and the tunnel attack require a strong encryption mechanism. Although, a one-way hash chain, is assumed to be present to solve these attacks; practical implementation of actual OHC can be adopted in future work e.g. to check message and computation overhead etc. Some DoS attacks, such as spoofing, false injection etc, are still partially dependent on encryption. Moreover, considerable research is being carried out to improve encryption in resource constrained nodes. This work might be incorporated in the newly proposed protocol.

The use of a global key (the encryption mechanism adopted in this research) also has some drawbacks and the protection of the global key as well as its authentication update for future node addition is an important challenge. The proposed scheme, which uses the global key to set up the keys between the neighbour nodes, inherits the same vulnerability as LEAP (i.e. it is vulnerable to attackers which remain active during the KSP). This is the motivation for completing KSP quickly, thus avoiding any node being compromised during that period. Therefore, when additional nodes need be added later, the global key will be non-existent at that time. μTESLA [38] is one possible solution that can be adopted for an authentic update. If the global key mechanism has to be retained, these weaknesses must also be addressed. Moreover, any other key management scheme can be used to address these issues; this research has not committed to a specific key management scheme.

The loud scheme presented as the initial wormhole/INA protection has also some limitations. A simple wormhole is unsuccessful in RAEED. A wormhole failing the encryption scheme and knowing that the signal coming is the LOUD message also remains unsuccessful. However, a problem arises if the attacker is able to measure the signal strength accurately and act accordingly by tunneling with the modified signal strength, since the scheme will fail. Although this process is very difficult to achieve with the random sensor deployment, it indicates that a weakness does exist in this scheme. The proposed protocol, however, solves this issue by employing a neighbourhood watch approach to prevent the links that do not the forward data. Thus any successful wormhole and INA links will be removed. But this drawback must be addressed before implementing the proposed loud scheme in any other routing protocols.

The ranking system used in the protocol needs further improvement. Different issues such as node balancing and loss of messages due to noise must be taken into consideration. As a result, data takes multiple paths if the message feedback is not eavesdropped and multiple copies of data are received at the BS. Although, this has increased the throughput it comes at the expense of an increased average hop count and additional power consumption. These multiple copies must be suppressed in future, in such a manner that the rushing attack is not possible by intruders.

The proposed new protocol, RAEED, has been verified empirically using practical implementations (a few attacks). This work can also be improved by implementing all DoS attacks considered to study real radio effects including fading, memory consumption, other hardware issues etc. Work can also be done to implement RAEED in different applications e.g. by implementing it in a forest, disaster relief situations, a battle field etc. Moreover, energy efficiency of RAEED can also be studied practically.

The methodology for the analysis of routing protocols, presented by this research can also be improved. Although, different formal model checkers have been studied before implementing the formal framework the model development is not automatic. Currently, researchers are implementing models directly from source code or vice versa without a human interference.

This type of work might be incorporated in the formal framework so that the model is developed directly from nesC [230] code. The conversion of event driven style of nesC to a formal modelling is a challenging task so another solution is required to enable the framework to convert the protocol specifications automatically to nesC code and Spin/Uppaal. A low fidelity simulator can be used to test networks with more than 1000 nodes. A higher level simulator with different radio models (NS2 simulator) can be used to assess the effects of different radio models and noise on larger networks. Finally, work may be extended to measure the energy efficiency of RAEED using both formal modelling and simulation.

Appendix A

Assumptions, Methods and Database

A.1 Introduction

This appendix is organized as follows: Section A.2 briefly describes the assumptions adopted when modelling networks (e.g. size of nodes, number of neighbours, etc.), radio links, cryptography, attacker capability and RF noise. Section A.3 briefly explains the formal modelling tool Uppaal used in the research with an example. The need of computer simulation in the research and adopted noise model is discussed in Section A.4 and Section A.5 briefly introduces the practical implementation. The alternative scheme for DFP, the Handshake scheme, is presented in Section A.6. Finally, the database required in RAEED is presented in Section A.7.

A.2 Assumptions

Certain assumptions are made concerning RF radio, networks, cryptography and attackers. These assumptions are important for verification and simulation. Most of these assumptions are the ones generally adopted by most researchers. Apart from these assumptions, which are generally applied in both simulation and formal model-checking, some additional assumptions are considered for formal model-checking which are discussed later in Section 4.3.3.

A.2.1 Network Assumptions

A.2.1.1 Node Placement and Topology

The topology of a WSN may change slightly at any time as some nodes may die or get damaged and new nodes may be added to the network. The nodes can be scattered (random placement) or placed manually in a rectangular grid. Experiments were performed using both types of node placement. Although a random topology imitates a real world scenario; grid placements are preferred because this will keep the effects of the topology constant while other factors can be

reproducibly studied. Random topologies are only used, when the effect of all other factors have been evaluated and the aim is to ascertain the effect of the topology on the routing protocol considered.

A.2.1.2 Network Size

A WSN may be composed of a large number of nodes. The current research has been restricted to a maximum of 1000 nodes. The reason being the simulator chosen has high fidelity so checking a big network becomes infeasible. The networks larger than this size may be the subject of future work.

A.2.1.3 Node Density

Node density is the average number neighbours each node possess in a network. WSN nodes are assumed to be operating in both high and low densities. Usually a WSN has high density ($\geqslant$20) but the scattered nature and unknown deployment might cause a WSN to have a low density. Moreover, power depletion and environmental effects might cause the densities to be reduced with time. In this research node densities of 4, 8, 12, 20, 24 and 28 square grid networks have been used. These values refer to non-boundary neighbour nodes in grid networks. The actual average density will be little less for each case because the boundary nodes in a grid network always contain fewer neighbouring nodes. Note that a density of 16 is not possible in a grid network if the radio range is considered symmetric.

A.2.1.4 Limited Resources

One of the most important properties of WSN nodes relates to their limited memory and computational power. Therefore, expensive hardware like GPS, tight time synchronisation (nanosecond precision) etc are considered unavailable. These assumptions are decisive when cryptographic scheme is chosen for message protection.

A.2.1.5 Powerful Base Station

The base station is assumed to be containing unlimited resources (e.g. a laptop). Also it is assumed that a BS cannot be compromised.

A.2.2 Radio Links Assumptions

A.2.2.1 Circular Links

Nodes are assumed to be having omnidirectional antennas so that data sent by a node will be captured by all neighboring nodes i.e. nodes within the radio range of the node. Unlike the real world, the RF range is considered as circular in computer simulation and formal modelling. Essentially, the radio model considered is a two-dimensional model of radio propagation with

circular range. The circular range is common practice in research using in computer simulation and formal modelling to avoid using complex RF models.

A.2.2.2 Bidirectional/Unidirectional Links

To reflect real world RF conditions, it is considered that links between nodes can be bidirectional or unidirectional. Previous research often has assumed links to be bidirectional. For the new protocol, it is assumed that the radio links might fade with passage of time. However, the neighbourhood watch scheme adopted will remove those weak links automatically.

A.2.2.3 Adjustable Radio Transmission Power

A node is assumed to be working in two different transmission modes. A *low power mode* in which data and other information is exchanged and a *high power mode*, in which the node increases the power of transmission to transmit some special messages. By using the high power mode a node's transmissions can be received by the nodes located two hops away. Note that the current MICAZ motes [233] provide this flexibility in the data transmission, and no extra circuit is required to fulfill this assumption.

A.2.2.4 Ideal Link Layer

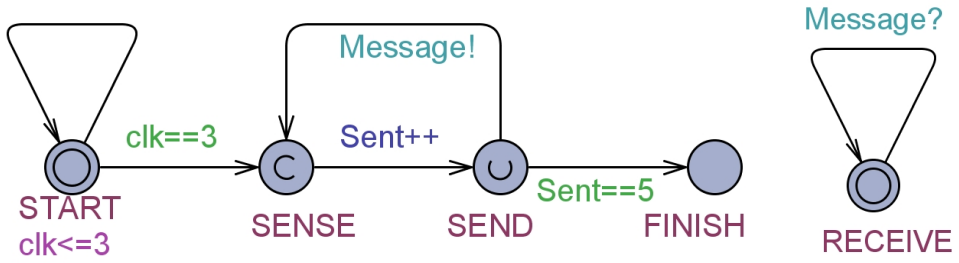
It is assumed that the link layer handles issues like data collision, RF channel access etc. However, the hidden terminal may cause the collision of messages.

A.2.3 Cryptography Assumptions

It is assumed that some variety of data encryption such as a message authentication code (MAC) is deployed. This encryption might be symmetric key cryptography (SKC), one way hash chain (OHC) or public key cryptography (PKC), depending on the application and the capabilities of nodes.

A.2.4 Attacker Assumptions

An attacker is assumed to be very powerful compared to normal sensor nodes; it can launch any attack on the WSN. The attacker can eavesdrop the keys if a weak encryption technique is applied and can capture/compromise a node if a strong encryption technique is used. This enables an attacker to launch an attack in all possible situations. Moreover, an attacker is assumed to be equipped with antennas enabling a long transmission range and unlimited power. Thus the attacker model employed here resembles the Dolev-Yao model [243]. Dolev and Yao define the attacker as: "someone who first taps the communication line to obtain messages and then tries everything it can to discover the shared secret" [243]. The Dolev-Yao model assumes that the attacker may eavesdrop all messages, the attacker is a trusted user and can initiate a



(a) Transmitter model

(b) Receiver model

Figure A.1: An example of Transmitter/Receiver system to explain Uppaal modelling

connection to any node. The current research is concerned with DoS attacks on WSN. Thus each DoS attacker is defined (Section 3.4) separately assuming the properties of Dolev-Yao model.

A.3 Formal Modeling Tool: Uppaal

In this section, we introduced here an informal description of Uppaal; more complete and fully formal description are available in [197]. Uppaal is extensively used for the verification of real time systems. The system is modelled using a timed automata(s), which is a finite-state machine extended with clock variables. A system state is defined as the automata location and values of clock and other variables. A automata may fire transitions synchronously (with other automata) or separately, leading to new states. There are three types of location in Uppaal: normal, urgent and critical. A *normal* location may or may not contain an *invariant* (a condition that must be true on clock or variable in a state). In an *urgent* location time is not allowed to pass but interleaving with other normal states is allowed. A *committed* location is a restrictive state enabling the next transition immediately. So of all the possible transition in the model the committed location must execute at all costs. In addition to enforcing particular behaviour, the use of urgent and committed locations reduces state space and improves time efficiency. The locations in Uppaal are represented using circles and transitions are represented using direct arcs. The urgent and critical locations are represented using U and C inside the circle. Each process (model) in Uppaal must have a initial location indicated using a double circle. The complete Uppaal model is a parallel composition of all the processes or models.

In summary, the system in Uppaal is composed of timed automata(s) called *templates*. The independent *models* are then implemented from these templates. All possible *states* are the cross products of all the model transitions and locations. To explain how Uppaal is used, a simple transmitter/receiver system is modelled as an example as shown in Figure A.1(a) and Figure A.1(b). This simple system consists of a transmitter that transmits five messages and the receivers receiving these messages. The two independent sub systems, as shown in the figures, are the templates. If one transmitter and two receivers are considered in the complete system then transmitter state model is developed from transmitter template and two receiver state models are developed from receiver template by Uppaal as a parallel composition. The transmitter model consists of 4 locations and 5 transition including a self transition (arc head and tail are attached to the same location). The receiver model, on the other hand, consists of only a single location and a self transition that is only triggered whenever it receives a message. Each model has an initial location indicated by double circle. When the system starts, all models are always in their initial locations. The messages exchanged in Uppaal may be unicast or broadcast; but in this thesis broadcast mode is chosen to model RF communication transition. The '!' indicates a messages transmitted and '?' indicates that a message is received. In case a message is broadcasted in Uppaal using '!' all models having '?' in active transmissions will receive this message.

The message exchanged in this model is a global channel variable, `Message`, which is transmitted by transmitter. The transmitter model starts in the START location. An invariant on local clock variable `clk` keeps the model in this location until the clock is updated to 3 ticks. The time passes automatically in Uppaal. Note a self transition is present in the START location that will execute until the clock value becomes 3. If this transition is not present the model will deadlock here. When the clock value becomes equal to 3 ticks a guard on the second transition moves this model to the SENSE location. This location is critical indicated by a C inside circle. This means a system cannot delay in the SENSE location and among all the available active states the only possible transition is from the SENSE location. This transition updates `Sent` variable which indicates the number of messages sent by transmitter. The next location, SEND, is urgent (U inside circle) which means time cannot pass when the system is in the SEND location. This is equivalent to having an extra clock c which is reset (c=0) before model enters the SEND location and presence of an invariant of c<=0 on this location. The model cannot move to the FINISH location until the value of `Sent` variable becomes equal to 5 (guard at this transition) so the only possible transition is moving back to the SENSE location and transmits `Message`. The `Sent`, which is a local variable, is incremented each time and when its value becomes equal to 5 the model moves to the FINISH location and deadlocks there.

The main purpose of the Uppaal model-checker is to verify that the model satisfies the required specifications. These specifications are expressed in a formally defined query language based on CTL (computation tree logic). The query language consists of theorems (claims) which

are either state formula (which describe individual states) or path formula (which quantify over paths or traces of model). "A state formula is an expression that can be evaluated for a state without looking at the behaviour of the model" [197]. The path formulas are further classified as safety, liveness and reachability properties. The reachability properties are simply sanity checks, the liveness properties are defined as 'something good will eventually happen' while the safety properties are defined as 'nothing bad will never happen'. Thus the data which never reaches the sink can be termed as a safety violation. The protocol deadlock is termed as the liveness failure. A data generated by a source node eventually reaching the sink is also termed as liveness check. The path formulas supported in Uppaal are A[] Φ, A<> Φ, E[] Φ, E<> Φ, $\Psi \rightsquigarrow \Phi$ and $\Psi \Rightarrow \Phi$. Here Φ and Ψ are state formulas. Whereas in Uppaal E and A indicate 'eventually' and 'always', respectively, while <> and [] are symbols for 'one path' and 'all paths' respectively. Other important symbols used in properties are $\Rightarrow$ and $\rightsquigarrow$ meaning implies and leads to respectively. More details about these properties can be found in Uppaal manual [197]. For example in the Receiver/Transmitter system a property of interest is that eventually the transmitter model moves to the FINISH location. This can be expressed formally as:

$$E <> Transmitter.FINISH$$

A.4 Computer Simulation

Computer simulators can simulate WSNs comprising thousands of nodes. Modern tools and low level details (high fidelity with the actual software) implementation in node's design in most computer simulation enables debugging/checking easy to handle. Moreover, one can run a simulation at a convenient speed, or even pause a simulator and thus can check all the results desired, this is not feasible when run time checking during physical implementation. Simulation, however, no matter how accurately they are developed, are not physical experiments; it is quite possible that a WSN may work perfectly on a simulator but fails if implemented. Moreover, they also cannot perform exhaustive testing possible using formal model-checkers. Most of the ongoing research efforts in the development of simulation tools focus on the study of two main WSN characteristics:

- New protocols and control mechanism to facilitate fast configuration and management of WSN. The aim is to prove and validate the specific protocols of MAC and routing.
- Development of tools capable of emulating the actual behavior of wireless nodes and the main aim is to check implementation issues such as energy, usage etc.

A.4.1 Simulator Selected: TOSSIM

Most simulators are aimed at one of these two objectives. Many simulators have been developed to check wireless routing protocols. Some aim to be versatile allowing both wired

and wireless medium to be examined like Ns-2 [244], GloMosim [245] and OmneT++ [246]. Some simulators aim for high fidelity in wireless nodes like ATEMU [247], Avrora [248], JSim [249] and TOSSIM [229]. Some are focussed on sound propagation and acoustics like Visual Sense[250] and SENS [251]. Some aim for high scalability and performance like Shawn [252] and TOSSF [253]. Some implement accurate radio models like Powler [254] and JProwler [255]. No simulator is, however, perfect in all fields. Of all these simulators TOSSIM and Ns-2 provide features of use here and are most widely adopted by WSN researchers.

The *Ns2 simulator*, although widely used, but it has some shortcomings. It requires a lot of learning, has given more attention is given to TCP and UDP (wired) protocols instead of wireless protocols, object-oriented design introduces quite an unnecessary interdependency between modules making the addition of new protocol models extremely difficult, and does not have high fidelity for wireless simulations in spite of providing highly detailed packet level.

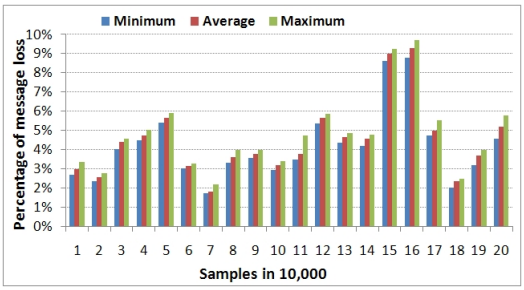
The simulator used in research, *TOSSIM* [229], operates at a very low level. It is implemented using C that provides high level of accuracy (bit level) by using models of only a few low-level components and otherwise running the source code unchanged. However, the nodes must be Crossbow AVR processors or MICA2 motes (also known as motes). TOSSIM compiles nesC [230] source code together with TinyOS [2] operating system libraries into a binary code for the development workstation, replacing the software modules that interface hardware with emulation libraries, including timers, communication channels, sensors, and the radio. TOSSIM compiles code written for TinyOS to an executable file that can be run on a standard PC. Using this techniques developers can test their implementation without using hardware motes.

TOSSIM was not developed with high scalability in mind and could initially simulate networks of up to 1000 nodes. This was later increased up to 8,192 nodes. TOSSIM can measure packet losses, packet CRC failure rates, and the length of the send queue during simulation. TOSSIM is not as accurate as other high fidelity simulators like ATEMU [247] and Avrora [248]. Moreover TOSSIM uses an inaccurate probabilistic bit error model for the wireless medium. Also TOSSIM does not simulate the Mote's devices (Digital I/O and A/D) it just generates a random number for ADC etc. TOSSIM enables the simulation of sensors and actuators; however, it does not simulate the physical phenomena that are sensed.

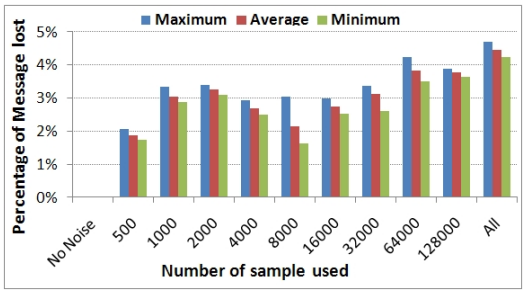
A.4.2 Aims of Simulation in the Thesis

Computer simulation allow the analysis in this thesis to be extended in a number of ways:

- It can confirm the existence of bugs discovered theoretically by model-checking.
- It provides some quantitative insight into the likely detrimental effects of bugs.
- It offers some measure of confidence in the robust behaviour of protocols in networks which are too large for model-checking.



(a) Using sample sizes of Meyer



(b) All samples of 10,000 from Meyer Noise Model

Figure A.2: Percentage of message lost due to noise

In this research, the experiments are usually performed 20 times and average values noted. In case when the effect of a certain factor was to be observed (Appendix A), the experiments are reduced to 10 (or below). In case the results obtained vary a lot between maximum and minimum values, additional experiments up to 30 are performed.

A.4.3 Noise in Computer Simulation

RF Noise is always present in the real world. The current research also aims to test the effect of noise. TOSSIM's radio model are based on the CC2420 radio (MICAZ etc). It uses an SNR curve derived from experimental data collected using two MICAZ nodes. TOSSIM uses the Closest Pattern Matching (CPM) algorithm which takes a noise trace as input and generates a statistical model from it. The Meyer-heavy noise trace is normally used in TOSSIM to model noise and it is believed to be accurately represent a noisy WiFi environment [256]. However, its use consumes a lot of memory (10MB per node) and so is not feasible when large networks are studied. A possible solution could be to sample the file with the aim of simply model the effect of noise (loss of some messages due to noise). To choose a suitable sample the research performed two different types of experiment. First the Meyer file is divided into blocks of 500 lines (500, 1000, 2000, 4000). Second the whole Meyer file is divided into 20 samples of 10,000 lines. For both cases, 20 tests were performed and then average, minimum, and maximum percentage message loss were calculated. The results are displayed in Figure A.2(a) and Figure A.2(b) respectively.

The effect of message lost due to noise was measured at different nodes and it was observed that, although the percentage of message loss differed, the pattern of message loss remained the same. The results presented are those obtained on a single node. It was observed that by using different samples of 10,000, different noise percentage loss can be achieved. The Meyer file includes peak values of noise (evident in sample 15, 16). On the other hand there are some

very low noise periods (sample 7, 18). These different samples are used in the current research to simulate effects of high and low noise. A further discussion on these noise samples appears in Section 5.5.4.2.

A.5 Practical Implementation

Some parts of the newly developed protocol were implemented in hardware to confirm that the protocol can be implemented practically and investigate other issues such as memory size, real radio effects etc. The hardware used was MICAZ motes [233] equipped with a 4Mhz Atmel microprocessors with 4 KB of RAM and 128 KB of code space, a 2.4 GHz radio running at 50 kb/s, 512kB of EEPROM and two AA batteries. The network size was limited (a maximum of 10 nodes). The in-built LEDs and an external buzzer board are used to provide debugging information. However, this implementation was limited and much remains as future work.

A.6 DFP: The Handshake Scheme

As stated earlier, this thesis presents two different schemes for the DFP: Lost Indication Scheme presented earlier in Section 5.7.1 and Handshake scheme that is presented in this section.

A.6.1 Design of the Handshake Scheme

A node N upon detecting an event in the environment generates the DATA that contains the event or message ID, node's own ID and a source ID (saved in TargetID field). The source node also attaches a data stamp in HMAC field used later by the BS to confirm that the data has not been altered in the relay process by any node (integrity). Moreover the data is encrypted with node's Independent key (authenticity) so that only BSs can decrypt what the actual data is (confidentiality). The other nodes just relay/forward this encrypted data without knowing what that data is. This provides data authentication, integrity and confidentiality from source to the destination. The message is encrypted by the source node S using the cluster key K_{C_S}:

$$N \rightarrow * : (DATA, [ID_S, ID_S, MID_m, Stamp_S, [datapayload]_{K_{I_S}}]_{K_{C_S}}) \tag{A.1}$$

A node N upon receiving data from the source node S modifies the message (replace node ID) before forwarding:

$$N \rightarrow * : (DATA, [ID_N, ID_S, MID_m, Stamp_S, [datapayload]_{K_{I_S}}]_{K_{C_N}}) \tag{A.2}$$

On receiving this DATA message, the neighbour node R unicasts back ACCEPT beacon if it is free and is willing to forward the data. This is a unicast message and the target field contains the sender's ID. The Source ID and Message ID are also attached for which the response has been sent. The message format is as follows:

$$R \rightarrow N : (ACCEPT, [ID_R, ID_N, -, -, -, -, -, ID_S, MID_m]_{K_{C_N}}) \tag{A.3}$$

On receiving ACCEPT beacons, the node selects a neighbour based on its previous ranking and broadcasts the DATA as well as it broadcasts SELECT beacon containing all the selected neighbours ID that are chosen for data forwarding:

$$N \rightarrow * : (SELECT, [ID_N, -, -, -, -, -, -, ID_S, MID_m, ID_1, ID_2, ID_3, ID_4]_{K_{C_N}}) \qquad \text{(A.4)}$$

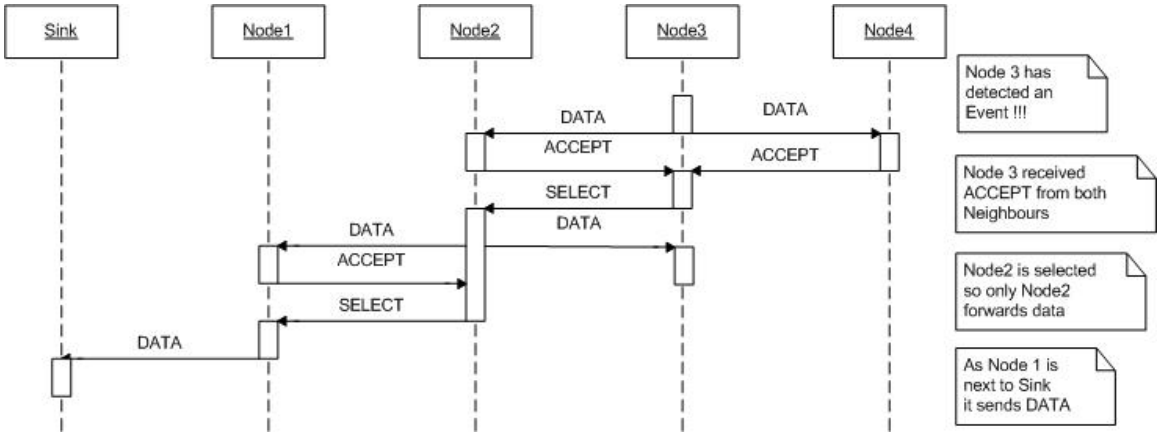


Figure A.3: Message sequence diagram to describe DFP (Handshake scheme)

Note that each ID can also indicate the BS if multiple BSs are used e.g. if 4 BSs are used the first ID means data must be forwarded to the first BS etc. Also note that the BS information is preloaded in the nodes before deployment. And that a node one hop distance away from the BS does not need to perform this process, it merely broadcasts the DATA to that BS instead. This complete scheme is explained in the message sequence diagram (Figure A.3).

A.6.2 Comparison of the Handshake Scheme with Flooding

It is evident that DFP has 2 additional messages for handshake apart from the DATA message. So the protocol will use more bandwidth and energy than the single data broadcast. For confirmation, the results are compared with the worst case i.e. flooding of data.

A.6.2.1 Visual Inspection

Let us suppose a network has a size of N nodes, density D, number of BSs is B and hop distance from a BS is H. As a shortest route is normally selected in the absence of attack, so H is the minimum hop distance from all the available BSs B. Let the messages be denoted by M_x where x is data (D), accept(A) or select(S). Then for flooding the total messages will be:

$$Total = N \times M_D \tag{A.5}$$

For the new protocol, when a node broadcasts the data, each neighbour node will send ACCEPT beacon and is finally followed by the node sending the Select beacon. So for each data packet the total messages are:

$$Total = M_D + (M_A \times D) + M_S \tag{A.6}$$

Now in the protocol's case as messages are not flooded, so the number of messages will be depending upon the hop distance from the BS:

$$Total = (H - 1) \times (M_D + (M_A * D) + M_S) + 1 \tag{A.7}$$

Note that it will take H-1 hops for the data to reach a node next to BS, which will forward data directly to BS and thus handshake will not be required at that stage. Now if there are B BSs at corners then the hop count is reduced by B times:

$$Total = \frac{H - 1}{B} \times (M_D + (M_A \times D) + M_S) + 1 \tag{A.8}$$

The hop count H is inversely proportional to density D of the network i.e. hop count decreases as the density is increased. So multiple of $H \times D$ will almost be the same for all densities. For comparison of results, different size & density networks are compared in the Table A.1. As the results are tabulated for the worst cases, only one BS is used. In case of multiple BSs, the results will improve for the new protocol but not for flooding (data will always be flooded to all nodes).

Note that the table containing the tabulated results for the new protocol are for the worst cases (nodes that are farthest from the BS), nodes in the middle and the best cases (nodes next to the BS). The total messages will be an average of all the nodes. Hence, the overall result will improve. Whereas for flooding, even the one-hop nodes will flood the data. Therefore the results will remain the same for all cases no matter how far away a node may be. Thus for N nodes network a total of N^2 data messages have to be propagated. Whereas for the new protocol the total messages will never reach N^2. As the results of protocol are worse for small networks, this thesis confirms the results using a 25 node network by employing a formal model.

Table A.1: Parameters used to check the effect of Level propagation delay on RSP

Nodes (N)	Density (D)	Hop count Distance (H)	Flooding Results	Protocol Farthest	Middle	OneHop
100	3.63	18	100	96.71	48.85	1
100	6.91	9	100	72.28	36.64	1
100	10.14	9	100	98.12	49.56	1
100	15.96	6	100	90.8	45.9	1
100	18.54	5	100	83.16	42.08	1
100	21.34	5	100	94.36	47.68	1
1000	3.88	62	1000	359.43	180.22	1
1000	7.63	31	1000	289.89	145.44	1
1000	11.37	31	1000	401.98	201.49	1
1000	18.64	21	1000	413.77	207.38	1
1000	22.15	16	1000	363.25	182.13	1
1000	25.74	16	1000	417.16	209.08	1

A.6.2.2 Formal Modelling

The hypothesis for this model is that:

"The new protocol's data propagation has less message overhead than flooding of data".

The assumptions stated in Section 4.3.3 are persisted in this model. Moreover it is assumed that the protocol has successfully finished the RSP i.e. node levels have been assigned and known.

The **model** used comprises 2 parts, a sink model and node model. The ***sink model*** is shown in the Figure A.4. The model remains in the LISTEN location and receives data packets until all nodes have sent their data packets. A guard (`NodeCount==MAXNODE-1`) then takes the model to the FINISH location.

The ***node model*** is shown in the Figure A.5. The node model starts from the LISTEN location. If no node model is sensing the data (`Sensing` flag is false), the model senses the data and sends it (SEND_DATA). If the node is one hop away from the sink (node level is 1), it will send the data and then goes back to the LISTEN location. Otherwise the node will wait and receive Accept beacons (REC_ACCEPT). When all neighbour nodes have sent the Accept messages (`BusyNodes` becomes 0), the node transmits Select beacon (SEND_SELECT). Using a self loop in REC_ACCEPT location, the best neighbour is saved in `SavedID` variable. After sending Select beacon, the node clears global flag `Sensing` so that the other nodes may also send their sensed data and then it moves to the LISTEN location.

Note that sending sensed data at the same time is avoided to check the worst case and to

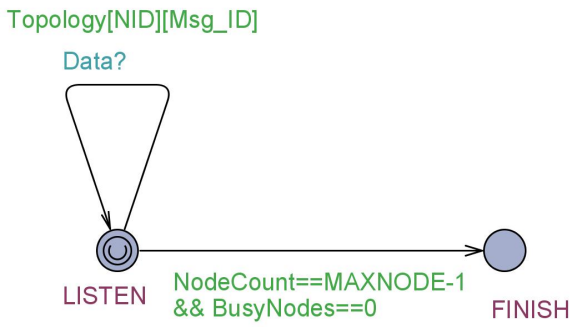


Figure A.4: Sink model used in UPPAAL to compare results with flooding

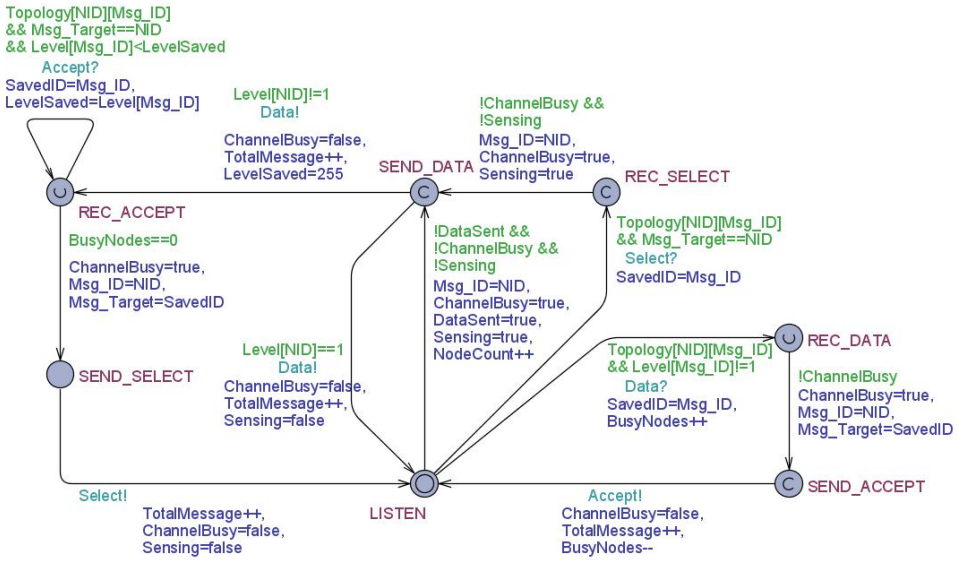


Figure A.5: Node model used in UPPAAL to compare results with flooding

simplify the model. A node upon receiving a data (REC_DATA) replies with an Accept beacon (SEND_ACCEPT). When a node has received a Select beacon (REC_SELECT), it tries to send data by moving to the SEND_DATA. Note also that the `ChannelBusy` flag is used to prevent 2 nodes transmitting any message simultaneously. Whenever a node transmits any message a global variable `TotalMessage` is incremented, that will indicate total messages transmitted by all nodes in the network. While the `NodeCount` variable is incremented each time a node transmits its sensed data.

The **verification** involves following claims to be proved using properties:

Claim 1: All nodes forward their own or other node's data

This is a sanity check in that all nodes will eventually either send their own sensed data or

another node's data:

$$E <> Node_{ID}.SEND\_DATA \tag{A.9}$$

Claim 2: All nodes will transmit their sensed data

This is a sanity check in that all nodes will eventually send their own sensed data:

$$E <> Node_{ID}.DataSent \tag{A.10}$$

Claim 3: The nodes are always available for data forwarding

This is a sanity check that all nodes after sensing or forwarding data will always go back to the LISTEN location to later perform further data forwarding.

$$Node_{ID}.SEND\_DATA \rightsquigarrow Node_{ID}.LISTEN \tag{A.11}$$

Claim 4: The BS will receive data packets from all nodes

This is a check that eventually the BS will receive data packets from all nodes in the network. The property used to prove this liveliness check is:

$$E <> Sink.FINISH \tag{A.12}$$

Claim 5: Total messages transmitted by new protocol are always less than flooding

This is a safety check that the total messages transmitted by protocol are always less than the case where flooding is employed. The property used for this claim is further extended to find what the worst case is, i.e. what is the maximum number of messages sent by a network in a worst case. For a 9-node grid network, this value was 37 for 8-neighbour network and 59 for 4-neighbour network. This is lesser when compared with flooding (64).

$$A[]\, TotalMessage <= N \tag{A.13}$$

Note that in the model the flooding value considered is 64 instead of 8 and in visual inspection the total flooded messages were equal to network size which is 8 in this case. The reason for the increase in number of total messages is because in the visual inspection only one node was considered as the source; whereas in the formal modelling all nodes are the source nodes. So each node will send 8 messages and thus a total of 64 (8x8). This means that total messages sent in data forwarding, when flooding is employed, are N^2 instead of N , where N is number of nodes excluding BS. So the formal model gives complete, accurate and better results as compared to visual inspection.

A.7 Database Required in the New Protocol

As indicated in the earlier sections, the new protocol maintains a small data base (DB) in each individual node. Apart from saving early information in KSP and RSP, the DB is required to store DFP information such as the neighbour's performance, data payload for sometime,

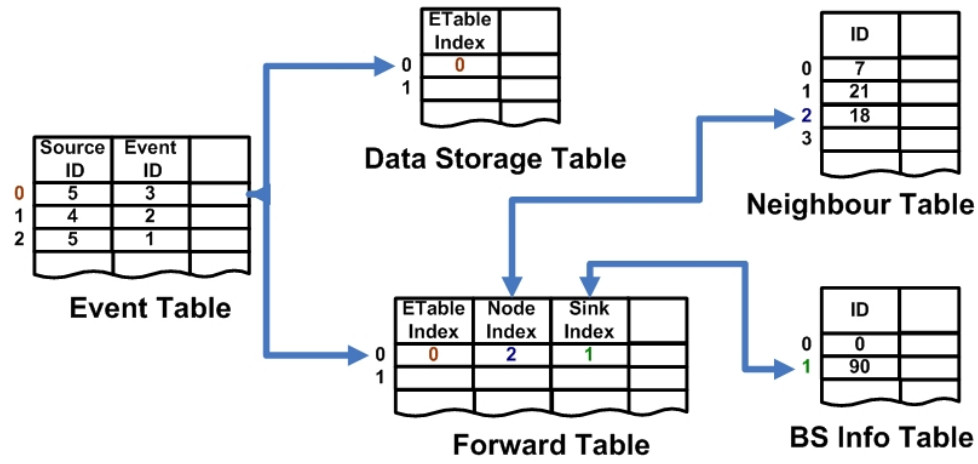


Figure A.6: Database Interface Diagram

and events/source information, etc. This DB can easily fit into WSN's limited memory. Five different tables have to be maintained by each individual node. These are Neighbour Table (NT), BS information Table (BT), Event Table (ET), Data storage Table (DT) and Forward Table (FT). The combination of these tables is a DB because all these tables are interrelated as indicated by Figure A.6. The FT and DT contain a unique key ETable Index, which is the address of ET. So entries in both these tables are related to an ET entry. Similarly, two fields in FT Node Index and Sink Index are the addresses of NT and BT respectively. The tables are discussed as follows:

A.7.1 Neighbour Table (NT)

The NT contains all the information about the neighbour nodes. In order to limit the effect of a sybil attack the maximum number of allowed neighbours can be fixed, while discarding the remaining. The entries in NT are created in the KSP when the BVP is performed. Nonce is also temporarily stored in the NT. The entries in the table are:

- ID, is unique ID of a neighbour node.
- Pair Key, for that particular neighbour node.
- Cluster Key, used by this neighbour node.
- Rank, is the number of messages, eavesdropped from the neighbour after the data has been forwarded to it. Initially, a nonce for that neighbour node is also saved in this field,

which is cleared after the KSP. Moreover if required the 2-hop Stamp is also saved in the RSP during the LPP.

- TotalSent, is the total number of messages forwarded to a neighbour. In the KSP, this field is used to save the number of attempts made to verify neighbours. Note that this provision is available in the new protocol to accommodate the loss of messages in the BVP.
- Lost, is the total data messages a neighbour node is unable to forward further and informed of this issue using LOST beacon (DFP).
- AskRec flag, is set when a node receives an ASK beacon from a neighbour node and is cleared when an ASSIGN beacon is sent back.
- Verified flag, is cleared initially and is set when a node receives back an ASSIGN in reply to an ASK (KSP).
- AssignSent flag, is set when a node sends ASSIGNACK beacon to that neighbour node. This will confirm that the 3 way verification (explained earlier) has been completed. After that the node will ignore future ASK beacons from this neighbour node.
- Level, is an array containing the neighbour level with respect to a partcular BS. This is updated in LPP (RSP).
- TwoHop, is an array of flags indicating the 2 hop neighbors witin radio range of the current neighbour. This entry is updated at the NPP (RSP) and is used later in DFP to confirm that a neighbour has sent data to a legitimate node rather than having been lost in oblivion.

A.7.2 BS Information Table (BT)

The BT maintains information regarding the BSs. A complete description of different fields is:

- ID, which is the BS's ID and is stored before node deployment.
- HashChain, which is hash chain stored for individual BS that should be used to later authenticate BS messages. This hash chain is updated with passage of time.
- Level, is a node's level (hop distance) with respect to a particular BS.
- Terror, is the time difference indicating when the node first receives LEVEL as compared to BS Authentic time.
- Received, is a flag used in the LPP (RSP) indicating that a LEVEL message for a BS has been received and has not been rebroadcasted as yet.

- Synchronize, is a flag used in the SPP (RSP) indicating that a SYNCHRONOUS message has been received and has not been rebroadcasted as yet.
- MustSynchronize, which is a flag that becomes true when the SYNCHRONOUS message is received and remains so until the LEVEL message is received.
- TimeMSec, is the time saved for a particular BS in one tenth of millisecond when LEVEL message is received.
- TimeSec, is the time saved for a particular BS in seconds upon receiving LEVEL message.
- TimeMin, is the time saved for a particular BS in minutes upon receiving LEVEL message.
- HMAC, is the HMAC saved for a particular BS when level message is received.

A.7.3 Event Table (ET)

The ET is used in DFP to save the (source,event) pair and the neighbour nodes to which that pair has been sent. Note that any sensed data from the environment has a unique pair to differentiate between them. The fields used in this table are:

- SourceID, is a unique source node's ID attached to a DATA message.
- EventID, is a unique event ID attached to a DATA message.
- Neighbors, is an array of flags representing respective neighbour nodes, to whom this event has been sent. Note that a one bit flag is used instead of complete neighbour ID, which is index to a NT.

A.7.4 Data storage Table (DT)

The DT temporarily stores the data in the node which has been forwarded to another node. This is done so that in case a neighour node fails to forward data, the saved data can be forwarded to another neighbouring node.

- ETableIndex, is the index or address of Event Table.
- DataStamp, is the data stamp which is stored.
- Data, is the complete data payload which has been saved.
- BSIndex, is the index or address of BT, indicating to which BS this data has been forwarded.
- HasSent, is a flag indicating whether the node has sent this data or is still waiting to send.

A.7.5 Forward Table (FT)

The FT maintains the data forwarding information that indicates to which node data has been forwarded.

- ETableIndex, is the index or address of Event Table.
- NTableIndex, is the index or address of Neighbour Table.
- FeedBack, is a flag indicating that the feedback has been received i.e. if the neighbor has forwarded the data further.

Appendix B

Verification of Claims

B.1 Claims on Models of Different Routing Protocols

B.1.1 Arrive Routing Protocol Claims

Claim 1: All nodes will broadcast a level message

This is a sanity check to verify that all the nodes eventually broadcast a level message, where N is a nodes's ID. This claim must be verified for all the nodes in the network. The specification property in Uppaal is:

$$E <> Node_N.SEND\_LEVEL$$

Claim 2: The source node eventually senses and broadcasts data

This is a sanity check to verify that a source node eventually senses and broadcasts data at least once. The following property states this claim formally, where S is the ID of the source node:

$$E <> Node_S.SENSE\_DATA$$

Claim 3: The Arrive protocol will always finish its setup phase

This is a sanity check to verify that the protocol eventually will enter the Data Forward Phase. This also confirms that the protocol will always finish the setup phase. Formally the property is written as:

$$E <> Protocol.DATA\_FORWARD$$

Claim 4: The Arrive protocol will not deadlock

The final sanity checks that the protocol will not deadlock and will finish eventually i.e. it goes to the FINISH location. Note that the Uppaal model, however, does deadlock, when EG is in the FINISH location, because only a limited number of messages are sent by the source. If this condition is removed from the model, then the system will never move to the deadlock state and will keep on sending the messages for ever. However, this has to be prevented, since

the main aim is to check certain properties and avoid the state space explosion problem. The protocol moves to the FINISH location after the source node has sent the desired number of data messages (set it to be 4) and no node is in its data forwarding mode. The following property presents this claim formally:

$$E <> Protocol.FINISH$$

Claim 5: All nodes attain the correct level
This claim is a check that all the nodes get the correct level when the Arrive protocol is in the operational phase:

$$Protocol.DATA\_FORWARD \rightsquigarrow (NodeLevel[N] == ExpectedLevel[N])$$

Claim 6: Neighbours and parents are assigned correctly
This is a property verifying that a source node S has at least one of its parents or neighbours attaining a lower or equal level so that the source will send the data to this node after the setup phase. The Arrive protocol will send data to only a parent (low level) or neighbour (same level).

$$Protocol.DATA\_FORWARD \rightsquigarrow ((NodeLevel[N1] <= NodeLevel[S]) \; || \; (NodeLevel[N2] <= NodeLevel[S]))$$

Claim 7: The BS will receive at least one data message from the source
This is a property which checks that the BS will eventually receive at least one data message from the source:

$$E <> Sink.DataRec$$

Claim 8: The BS receives a number of data messages from the source
This is a data transport property to check that when the protocol completes the number of received data messages at the BS is more than N, where N≤`TotalSent`.

$$Protocol.FINISH \rightsquigarrow (TotalRec > N)$$

B.1.2 ARAN Routing Protocol Claims

Claim 1 "Nodes always rebroadcast the RDP message:"
This is a sanity check to confirm that all the nodes fulfil the task of the route discovery phase i.e. rebroadcast the received RDP message. This Uppaal notation for this claim is:

$$Node_N.REC\_RDP \rightsquigarrow Node_N.SEND\_RDP$$

Claim 2 "Nodes always rebroadcast the REP message:"
This is a sanity check to confirm that all the nodes rebroadcast the received REP message. The

property in formal notation is:

$$Node_N.REC\_REP \rightsquigarrow Node_N.SEND\_REP$$

Claim 3 "Legitimate node forwards the data fairly:"
This is a sanity check to confirm that all the legitimate nodes forward data fairly; i.e. these do not drop data unless the node is a malicious black hole. The property in Uppaal notation is:

$$(!Blackhole[N]\&\&Node_N.REC\_DATA) \rightsquigarrow Node1.SEND\_DATA$$

The property states that all the nodes N in the network will always rebroadcast the data unless it is a black hole node when the data message is received.

Claim 4 "No deadlock in the ARAN protocol:"
This claim checks that the protocol will always finish after it has started and it never deadlocks in between:

$$EventGen.START \rightsquigarrow EventGen.FINISH$$

Note that the event generator can only reach the FINISH location if all nodes including the target and source become idle; i.e. they stop transmitting anything after the source has initiated the RDP. A more potent proof can be performed by applying the following property:

$$A[]notdeadlock$$

This property verifies that all the models never deadlock. However, the ARAN protocol model does have a deadlock; so this property fails and the trace generated confirms that the model deadlocks with the event generator model in the FINISH location. This deadlock had been intentionally introduced to save the state space. To confirm that the deadlock occurs because of this deliberate introduction and not due to another cause, a self loop is placed in the EventGen.FINISH location. Once done, the property did hold confirming that the model never deadlocks after it has reached the FINISH location which was our intended final location of the system.

Claim 5 "Data from the source node always reaches the target node:"
This final and most important claim is the data transport property that when the system finishes, the target node will have successfully received the data:

$$EventGen.FINISH \rightsquigarrow Target.SUCCESS$$

B.2 Claims on the Models of RAEED Routing Protocol

B.2.1 Bidirectional Property Claim in RAEED

Claim 1: All nodes will finish the KSP
This claim is a check that eventually the protocol will get out of the KSP i.e. all nodes will

begin the next phase of the protocol:

$$E <> Node_N.ROUTE\_SETUP\_PHASE$$

The claim is only proved when all N nodes of the network are checked. This claim can also be proved by checking if the event generator moves to the ROUTE_SETUP_PHASE location, which is only possible if all nodes have finished their KSP:

$$E <> Protocol.ROUTE\_SETUP\_PHASE$$

Claim 2: When the KSP finishes, then all the nodes have correct verified neighbours
This claim states that when the protocol has entered the Route Setup Phase(RSP) (and thus all nodes have finished KSP), the nodes have correctly verified the neighbours. Note that a function has been written in Uppaal that will increment a global variable `Problems`, in case a verified neighbour is found different from that in the connection matrix. The following Uppaal property proves this claim, where N indicates all the nodes:

$$Protocol.ROUTE\_UPDATE\_PHASE \Rightarrow (Node_N.Problems == 0)$$

B.2.2 Effect of Noise and Collision on the KSP

Claim 1: All nodes eventually enter Phase 2
This is a sanity check that all nodes eventually move to the PHASE2 location and is proved using the following property:

$$E <> Protocol.PHASE2$$

Claim 2: All 3 echo messages can be sent by each node
This is a sanity check. Note that nodes send three echo messages instead of one. This is to ameliorate the effect of message loss due to noise. The following property can prove this claim:

$$E <> (Node_N.EchoSent == 3)$$

Claim 3: When nodes enter Phase 2 there is no unidirectional link in the NT
This claim is a liveness check that when protocol has entered PHASE2 location, there is no problem in the NT of any node. Thus no entry in the NT has a unidirectional link. The property used to prove this claim is:

$$Protocol.PHASE2 \rightsquigarrow (Node_N.Problems == 0)$$

B.2.3 Confirmation that Security Properties Hold

Claim 1: The protocol does not deadlock
This is a liveness check to confirm that the model never deadlocks:

$$A[]!deadlock$$

Claim 2: All nodes eventually complete KSP

This is a sanity check that all nodes eventually complete their Key Setup Phase i.e. the event generator moves to the FINISH location. This claim can be proved by the following property:

$$E <> EventGen.FINISH$$

Claim 3: Global key compromise does not allow the attacker to capture the other keys

This is a sanity check that capturing the global key alone does not mean that the attacker may captured the other keys as well. The node must broadcast the respective message (containing pair key etc), only then the attacker captures the key. This sanity test is proved if the following 2 properties are false for all N legitimate nodes:

$$Attacker.DECRYPT\_GLOBALKEY \rightsquigarrow Attacker.Pair[N]$$

$$(Attacker.DECRYPT\_GLOBALKEY) \rightsquigarrow Attacker.Cluster[N]$$

Claim 4: Global key compromise leads to all further pair keys being captured

This is a claim that after an attacker has decrypted the global key then any node N, upon broadcasting its pair key (moving to WAIT location), will lead the attacker to capture the pair key of that node. The property used for this claim is:

$$(Attacker.DECRYPT\_GLOBALKEY \&\& Node_N.WAIT) \rightsquigarrow Attacker.Pair[N]$$

Claim 5: Global key compromise leads to all further cluster keys being captured

This is a safety check. It claims that when an attacker has decrypted the global key, and any node N has broadcast its cluster key (and thus moving to the FINISH location), it will allow an attacker to capture the cluster key of that node.

$$(Attacker.DECRYPT\_GLOBALKEY \&\& Node_N.FINISH) \rightsquigarrow Attacker.Cluster[N]$$

B.2.4 Verification of SPP and LPP (RSP)

For the verification properties *N1* and *N2* indicate any 1-hop distant and 2-hop distant node from the BS respectively. Following claims have been verified using properties:

Claim 1: The RSP will finish

This is a sanity check that the protocol will finish the RSP. In other words the sink model will reach the DFP location, meaning that all nodes have been assigned their level. The property that can prove this claim is:

$$E <> Sink.DFP$$

Claim 2: The sink will be synchronized with all nodes

This test states that eventually the sink model will be synchronized with all the nodes N in all possible executions. The property proving this liveness check is:

$$E[]Sink.clk == Node_N.clk$$

Claim 3: The sink will initiate the LEVEL only after the node clocks have been synchronized
This claim states that when the sink model initiate LEVEL message the clock of sink has already been synchronized with all the node models clocks. Here N means all nodes.

$$Sink.INITIATE\_LEVEL \rightsquigarrow (Sink.clk == Node_N.clk)$$

Claim 4: The lower hop nodes will always receive the LEVEL beacon earlier
This claim is the liveness check. It states that whenever any node N2, 2-hop away from BS (Node3,Node4,Node6) receives LEVEL beacon, any 1-hop (from BS) node's clock (Node1,Node2) is always greater than 2-hop node clocks. Following property can prove this claim:

$$A[](Node_{N2}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N2}.delay$$

This property is then repeated for other 1-hop and 2-hop nodes.

Claim 5: The clock difference between two hop nodes is always less than the maximum time reserved for each hop
This is a liveness check. Note that time reserved between 2-hop nodes is 10 clock ticks in this model. Thus this claim can be proved by using two properties:

$$A[](Node_{N2}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N2}.delay + 10$$

$$A[](Node_{N2}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N2}.delay + 9$$

The first property should be false whereas the second property should be true. The two properties claim that 2-hop nodes (Node3,Node4,Node8), on receiving the LEVEL beacon from 1-hop neighbour (Node1,Node2), must receive it not more than 10 milliseconds earlier. Note that this property should be satisfied for all values between 0 and 9 but not 10 as 10 means that 1-hop nodes have received the LEVEL beacon 11 ms before 2-hop nodes which is not possible as the model has set the time to be 10 ms to send LEVEL beacon after receiving it.

Claim 6: The clock difference between nodes 3-hop away from the BS is always less than the maximum time reserved for them
This is a liveness check. It is similar to previous properties and can be proved by using two properties:

$$A[](Node_{N3}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N3}.delay + 20$$

$$A[](Node_{N3}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N3}.delay + 19$$

Again the first property should be false whereas the second property should be true. The two properties claim that the nodes 3-hop away from BS (Node5,Node7) when receive the LEVEL beacon from any neighbour node the difference of clocks between them and 1-hop node (Node1,Node2) should not be more than 20 ms.

Claim 7: The clock difference between nodes 4-hop away from the BS is always less than the maximum time reserved for them

This is a liveness check. It is similar to claim 5 and 6. It can be proved by using two Uppaal properties:

$$A[](Node_{N4}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N4}.delay + 30$$

$$A[](Node_{N4}.RECEIVE\_LEVEL) \Rightarrow Node_{N1}.delay > Node_{N3}.delay + 29$$

Again the first property should be false whereas the second property should be true. The two properties claim that the nodes 4-hop away from BS (Node6) when receive the LEVEL beacon from any neighbour node the difference of clocks between them and 1-hop node (Node1,Node2) should not be more than 20 ms.

Claim 8: All nodes get the desired correct level

This is a safety check that all nodes N acquire the correct level when the protocol enters the DFP. Converting time to integers is not possible in Uppaal (type checking error), so difference between the BS time and node time is used to confirm if nodes have achieved the correct level. Each node, including the BS, resets a clock called time when it broadcasts LEVEL beacon. As nodes broadcast message after delaying it by 10ms, so a node H hops away from BS and it must have a time difference of 10xH ticks. The property employed to verify this claim is:

$$A[](Sink.DFP) \Rightarrow (Sink.time == Node_N.time + TimeNode_N)$$

In this equation N is any legitimate node in the network and $TimeNode_N$ is a constant, the expected hop count of node N multiplied by 10. Thus for 1-hop nodes this constant value is 10, for 2-hop nodes 20 and so on.

B.2.5 Verification of Lost Indication Scheme (DFP)

Claim 1: The data is transmitted fairly

This is a sanity check that all the nodes will eventually either transmit their own sensed data or the neighbour node's data. The property used to prove this claim is:

$$E <> Node_N.SEND\_DATA$$

Claim 2: The nodes after broadcasting the data never deadlock

This is a liveness check in that all the nodes upon broadcasting the data will not deadlock and should move back to the LISTEN state:

$$Node_N.SEND\_DATA \rightsquigarrow Node_N.LISTEN$$

Claim 3: No deadlock in the RAEED protocol

This is the liveness check that the RAEED protocol will not deadlock. Thus the BS and all the

node models will finish. To prove this claim the following property must be false:

$$A[]!Sink.FINISH$$

To confirm the liveness property mentioned in the last equation the deadlock property is checked:

$$A[]!deadlock$$

This property should be false. The system should deadlock with the BS in the FINISH state. To reconfirm the liveness check, a self loop is placed in the sink model's FINISH location. Now the deadlock property is satisfied indicating that the system never deadlocks. Note that the sink model only moves to the FINISH location when all the nodes are finished.

Claim 4: The source node will always send the desired number of data messages
This is a check that, eventually, the desired number of messages will be sent by the source node:

$$Sink.FINISH \rightsquigarrow (TotalSent == MAXMESSAGE)$$

Claim 5: The source node will always receive the desired number of data messages
This is the data transport property claiming that when all the nodes are finished, the BS receives all the data packets generated by the source node:

$$Sink.FINISH \rightsquigarrow (Sink.TotalRec == MAXMESSAGE)$$

Another way to verify the data transport property is:

$$Sink.FINISH \rightsquigarrow (Sink.TotalRec == TotalSent)$$

Claim 6: The throughput of the data messages is always 100%
Note that the previous safety property can also be true if the source node does not send any data packet at all. To confirm that the throughput was 100% it is checked that when a finite number of messages (`MAXMESSAGE`) are broadcasted by the source node, the BS also receives the same number. Following property proves this claim:

$$Sink.FINISH \rightsquigarrow (TotalSent == MAXMESSAGE)\&\&(Sink.TotalRec == TotalSent)$$

B.3 Claims involving the Evaluation of RAEED Against the Attacks

B.3.1 Prevention Against the INA and Wormhole Attack

Claim 1: The ASK beacon is received by legitimate neighbours
This is a sanity check that when Node1 broadcast the ASK beacon it will always be received

by neighbours Node0 and Node2. It is then later checked for all nodes and their neighbour combinations.

$$Node1.SEND\_ASK \rightsquigarrow (Node0.REC\_ASK \&\& Node2.REC\_ASK)$$

Claim 2: The protocol will finish the KSP

This is a sanity check that eventually the protocol will get out of the KSP, i.e. all nodes will go to the next phase of the protocol:

$$E <> EventGen.KSP\_FINISH$$

Claim 3: The protocol will finish RSP

This is a sanity check that eventually the protocol will get out of the KSP and RSP, i.e. the RSP will finish:

$$E <> EventGen.RSP\_FINISH$$

Claim 4: No deadlock in the system

This is both a sanity and the liveness check. It checks that all the nodes eventually finish and does not remain deadlocked.

$$E <> Node_N.FINISH$$

A more potent liveness check is that the protocol does not deadlock:

$$A[]!deadlock$$

This property should be false. The system should deadlock with the event generator and all the node models in their FINISH locations. To reconfirm the liveness check, a self loop was placed in EG's FINISH location. Now the deadlock property is satisfied indicating that the system never deadlocks. Note that the model only moves to the EventGen.FINISH location when all nodes are finished.

Claim 5: No legitimate node is declared as a wormhole

This safety check confirms that nodes attached to wormhole tunnels 0 and 4 do not detect any legitimate node as a wormhole. This test is proved by using following property:

$$A[](EventGen.FINISH \Rightarrow !Node0.Verified[1])$$

Claim 6: All the wormhole tunnels are detected

This is a safety check which confirms that a wormhole attack is detected by both nodes 0 and 4:

$$A[](EventGen.FINISH \Rightarrow Node0.Verified[4])$$

$$A[](EventGen.FINISH \Rightarrow Node4.Verified[0])$$

Claim 7: All wormhole tunnels are removed by the LOUD scheme

This is the safety test and checks that both neighbour nodes of the wormhole will have no neighbour error when the node finally goes to the FINISH location. Note that a flag, `Error`, remains set if the topology information updated in node is different from the `Topology` matrix. The following properties prove this safety test:

$$A[](EventGen.FINISH \Rightarrow !Node0.Error)$$

$$A[](EventGen.FINISH \Rightarrow !Node4.Error)$$

Claim 8: All the nodes record their correct neighbours after the LOUD scheme

This is a liveness check that all the nodes (N=1,2,3) which are not neighbours of the wormhole will have no incorrect neighbour i.e. no legitimate neighbour is removed because of loud test.

$$EventGen.FINISH \rightsquigarrow !Node_N.Error$$

B.3.2 Prevention Against the Intelligent Wormhole Attack

The claims 5 and 6 have been modified for this model because the flag `TestFail` is used to indicate a wormhole is detected by the node model. The modified claims are described again:

Claim 5: No legitimate node is declared as a wormhole

This is a safety check and confirms that the nodes attached to the wormhole tunnels 0 and 4 do not detect any correct node as a wormhole. This is proved by using following property:

$$A[](EventGen.FINISH \Rightarrow !Node0.TestFail[1])$$

Claim 6: All wormhole tunnels are detected

This is a safety check confirming that a wormhole attack is detected by both the nodes 0 and 4. This claim is proved using following two properties:

$$A[](EventGen.FINISH \Rightarrow Node0.TestFail[4])$$

$$A[](EventGen.FINISH \Rightarrow Node4.TestFail[0])$$

B.3.3 Prevention Against the Sinkhole Attack

Claim 1: Attacker's SYNCHRONOUS messages will always be rejected

This is a claim that when an attacker broadcasts SYNCHRONOUS message, all legitimate nodes will reject it. This claim is proved if the following property is false:

$$Attacker.FAKE\_SYNCHRONOUS \rightsquigarrow (Node_N.REC\_SYNCHRONOUS \&\& Msg\_ID == A)$$

In this equation N is any legitimate node ID and A is the message ID of the attacker.

Claim 2: Legitimate node's SYNCHRONOUS messages will always be accepted

This is a claim that when a legitimate node broadcasts the SYNCHRONOUS message, all legitimate neighbour nodes will accept it. This claim is proved if the following property holds true:

$$Node_N.SEND\_SYNCHRONOUS \rightsquigarrow (Node_M.REC\_SYNCHRONOUS \&\& Msg\_ID == N)$$

In this equation N is the node ID of any legitimate sender node and M is ID of the neighbour of node N.

Claim 3: Attacker's LEVEL messages will always be rejected

This is a safety check. It claims that when the attacker broadcasts the LEVEL message, all the legitimate nodes will reject it. This safety test is proved if the following property is false:

$$Attacker.FAKE\_LEVEL \rightsquigarrow (Node_N.RECEIVE\_LEVEL \&\& Msg\_ID == A)$$

In this equation N is any legitimate node and A is the message ID of an attacker.

Claim 4: Legitimate node's LEVEL messages will always be accepted

This is a liveness check. It claims that when a legitimate node broadcasts the LEVEL message, all legitimate neighbour nodes will accept it. This liveness check is proved if the following property holds true:

$$Node_N.SEND\_LEVEL \rightsquigarrow (Node_M.RECEIVE\_LEVEL \&\& Msg\_ID == N)$$

In this equation N is the node ID of any legitimate sender node and M is the node ID of node N's neighbour.

Bibliography

[1] J. Kahn, R. Katz, and K. Pister. Emerging challenges: Mobile networking for smart dust. pages 188–196. J. Comm. Networks, 2000.

[2] Tinyos. *http://www.tinyos.net/*, 2007.

[3] S. Madden. Tinydb:a declarative database for sensor networks. *http://telegraph.cs.berkeley.edu/tinydb*, 2003.

[4] J. Mulder, S. Dulman, L. van Hoesel, and P. Havinga. Peeros - system software for wireless sensor networks. *http://wwwhome.cs.utwente.nl/ dulman/docs/systemsoft.pdf*, 2007.

[5] Pumpkin. Salvo : The RTOS that runs in tiny places. *http://www.pumpkininc.com/*, 2007.

[6] Unicorn Web Services. Cmx systems. *http://www.cmx.com/*, 2006.

[7] R. Barr, J.C. Bicket, D.S. Dantas, B. Du, T.W.D. Kim, B. Zhou, and E. Sirer. On the need for system-level support for ad hoc and sensor networks. Number 2, pages 1–5, New York, NY, USA, 2002. ACM Press.

[8] Palm os. *http://www.palmsource.com/*, 2007.

[9] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. technical reports:ACM SIGARCH Computer Architecture News 30, 2002.

[10] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks (Elsevier) Journal*, pages 393–422, 2002.

[11] D. Estrin, R. Govindan, J.S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[12] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Journal on Wireless Communications*, 11:6–28, 2004.

[13] A.D. Wood and J.A. Stankovic. Denial of service in sensor networks. In *IEEE Computer*, volume 35, pages 54–62, 2002.

[14] G. Noubir and G. Lin. Low-power dos attacks in data wireless lans and countermeasures. In *Mobile Computing and Communications Review (IEEE MobiHoc)*, volume 7, pages 29–30, 2003.

[15] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2-3):293–315, 2003.

[16] J. Lopez and J. Zhou. *Wireless Sensor Network Security*. IOS Press, Fairfax, VA, USA, cryptology and information security series (cis) edition, 2008.

[17] J. Deng, R. Han, and S. Mishra. The performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *IEEE 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03), Palo Alto, CA, USA*, pages 349–364, 2003.

[18] J. Deng, R. Han, and S. Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, 2006.

[19] J. Deng, R. Han, and S. Mishra. INSENS: Intrusion-tolerant routing for wireless sensor networks. In *Elsevier Journal on Computer Communications, Special Issue on Dependable Wireless Sensor Networks*, volume 29, pages 216–230, 2005.

[20] T.R. Andel and A. Yasinsac. The invisible node attack revisited. In *Proceedings of IEEE SoutheastCon*, pages 686–691, 2007.

[21] L. Tobarra, D. Cazorla, F. Cuartero, G. Diaz, and E. Cambronero. Model checking wireless sensor network security protocols: Tinysec + leap. In *Proceedings of the First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07)*, pages 95–106. IFIP Main Series, Springer, 2007.

[22] L. Tobarra, D. Cazorla, and F. Cuartero. Formal analysis of sensor network encryption protocol (snep). In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS 2007), Piscataway, NJ, USA*, pages 767–772, Pisa (Italy), 2007.

[23] J. Guttman. Foundations of security analysis and design of lncs. In *Foundations of Security Analysis and Design*, volume 2171, pages 197–261. Springer, 2000.

[24] S. Yang and J. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *Proceedings of ACM Workshop Security of Ad Hoc and Sensor Networks*, pages 12–20, 2003.

[25] J. Marshall. An analysis of the secure routing protocol for mobile ad hoc network route discovery: Using intuitive reasoning and formal verification to identify flaws. Msc thesis, Dept. of Computer Science, Florida State University, 2003.

[26] Y. Hanna. Slede: lightweight verification of sensor network security protocol implementations. In *Proceedings of the doctoral symposium of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2007).Dubrovnik, Croatia*, pages 591–594, 2007.

[27] Y. Hanna, H. Rajan, and W. Zhang. Slede: a domain-specific verification framework for sensor network security protocol implementations. In *Proceedings of the first ACM conference on Wireless network security (WISEC '08), Alexandria, VA, USA*, pages 109–118, 2008.

[28] Y. Hanna and H. Rajan. Slede: Framework for automatic verification of sensor network security protocol implementations. In *Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), Vancouver, Canada*, pages 109–118, 2009.

[29] A. Gergely. *Secure Routing in Multi-hop Wireless Networks*. Ph.d. dissertation, Laboratory of Cryptography and Systems Security (CrySyS), Budapest University of Technology and Economics, 2009.

[30] L. Buttyan and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN), Washington DC, USA*, pages 94–105, 2004.

[31] G. Acs, L. Buttyan, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. In *IEEE Transactions on Mobile Computing*, volume 5, pages 1533–1546, 2006.

[32] G. Acs, L. Buttyan, and I. Vajda. Provable security of ondemand distance vector routing in ad hoc networks. In *Proceedings of ESAS2005, LCNS3813, Berlin*, pages 113–127. Springer, 2005.

[33] G. Acs, L. Buttyan, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *The Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2006), Alexandria, VA*, 2006.

[34] G. Acs, L. Buttyan, and I. Vajda. The security proof of a link-state routing protocol for wireless sensor networks. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, 2007 (MASS 2007)*, pages 1–6, 2007.

[35] L. Mao and J. Ma. Towards provably secure on-demand distance vector routing in manet. In *International Conference on Computational Intelligence and Security (CIS '08)*, volume 1, pages 417–420, 2008.

[36] M. Burmester, T. V. Le, and B. de Medeiros. Towards provable security for ubiquitous applications. In *Proceedings 11th Australasian Conference on Information Security and Privacy (ACISP 2006), Melbourne, Australia*, pages 295–312, 2006.

[37] T.R. Andel and A.Yasinsac. Automated evaluation of secure route discovery in MANET protocols. In K. Havelund, R. Majumdar, and J. Palsberg, editors, *Proceedings of 15th International SPIN Workshop on Model Checking Software (SPIN 2008), Los Angeles, CA, USA*, volume 5156 of *Lecture Notes in Computer Science*, pages 26–41. Springer, 2008.

[38] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. SPINS: Security Protocols for Sensor Networks. In *ACM Mobile Computing and Networking*, volume 8, pages 521–534, 2002.

[39] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA*, pages 22–31, 2002.

[40] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00)*, page 223, 2000.

[41] F. Ye, A. Chen, S. Liu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Proceedings of the tenth International Conference on Computer Communications and Networks (ICCCN), Scottsdale, AZ, USA*, pages 304–309, 2001.

[42] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[43] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *ACM Conference on Computer and Communications Security (CCS'03)*, pages 62–72, 2003.

[44] C. Karlof, Y. Li, and J. Polastre. Arrive: Algorithm for robust routing in volatile environments. Technical Report UCBCSD-02-1233, Computer Science Department, University of California at Berkeley, 2003.

[45] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP 02)*, pages 78–89, 2002.

[46] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A.Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. *Commun. ACM*, pages 272 – 287, 2001.

[47] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99) , Seattle, Washington, USA*, pages 174–185, 1999.

[48] S. Hedetniemi and A. Liestman. A survey of gossiping and broadcasting in communication networks. In *Networks*, number 4, pages 319–349, 1988.

[49] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *The International Journal of High Performance Computing Applications*, 16(3):293–313, 2002.

[50] R.C. Shah and J. Rabaey. Energy-aware routing for low energy ad hoc sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '02)*, volume 1, pages 350–355, 2002.

[51] N. Sadagopan, B. Krishnamachari, and A. Helmy. The acquire mechanism for efficient querying in sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, pages 149–155, 2003.

[52] C. Schurgers and M.B. Srivastava. Energy efficient routing in wireless sensor networks. In *Proceedings of IEEE Military Communications Conference on Communications for Network-Centric Operations (MILCOM): Creating the Information Force, McLean, VA*, volume 1, pages 357–361, 2001.

[53] S. Kim, S.H. Son, J.A. Stankovic, S. Li, and Y. Choi. Safe: A data dissemination protocol for periodic updates in sensor networks. In *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, pages 228–234, 2003.

[54] S. Lindsey and C.S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *IEEE Aerospace Conference Proceedings*, volume 3, pages 1125–1130, 2002.

[55] A. Manjeshwar and D.P. Agrawal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (WPIM 2002)*, page 195, 2002.

[56] A. Manjeshwar and D.P. Agrawal. Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of the*

International Parallel and Distributed Processing Symposium (IPDPS 2002), pages 195–202, 2001.

[57] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *ACM SIGMOD Record archive*, volume 31, pages 9–18, 2002.

[58] B. Karp and H.T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual ACM/IEEE international conference on Mobile computing and networking (MobiCom '00)*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[59] Y.-B. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 66–75, 1998.

[60] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MOBICOM '98), Dallas, Texas, USA*, pages 76–84, 1998.

[61] V. Rodoplu and T.H. Ming. Minimum energy mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 17(8):1333–1344, 1999.

[62] L. Li and J. Y Halpern. Minimum energy mobile wireless networks revisited. In *Proceedings of IEEE International Conference on Communications (ICC'01), Helsinki, Finland*, volume 1, pages 278–283, 2001.

[63] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01), Rome, Italy*, pages 70–84, 2001.

[64] Y. Yu, D. Estrin, and R. Govindan. Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA-CSD TR-01-0023, Computer Science Department, UCLA, Los Angeles, California, USA, 2001.

[65] W.H. Liao, Y.C. Tseng, K.L Lo, and J.P. Sheu. Geogrid: A geocasting protocol for mobile ad hoc networks based on grid. *Journal of Internet Technology*, 1(2):196–213, 2002.

[66] Shibo Wu and K. Selcuk Candan. Gmp: Distributed geographic multicast routing in wireless sensor networks. In *Proceedings of 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, page 49, 2006.

[67] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Journal of Personal Communications*, 7(5):16–27, 2000.

[68] H. Tian, J.A. Stankovic, L. Chenyang, and T. Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems 2003*, pages 46–55, 2003.

[69] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural. Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM 2005)*, volume 4, pages 2646–2657, 2005.

[70] K. Akkaya and M. Younis. An energy-aware qos routing protocol for wireless sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems 2003*, pages 710–715, 2003.

[71] F. Ye, S. Lu, and L. Zhang. Gradient broadcast: A robust, long-lived large sensor network. 2001. http://irl.cs.ucla.edu/papers/grab-tech-report.ps.

[72] J.H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. In *Proceedings of the Advanced Telecommunications and Information Distribution Research Program (ATIRP'2000),College Park, MD*, pages 22–31, 2000.

[73] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan. Upper bounds on the lifetime of sensor networks. In *IEEE International Conference on Communications (ICC 2001), Helsinki, Finland*, volume 3, pages 785–790, 2001.

[74] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. In *Communications of the ACM (2004)*, number 6, pages 53–57, 2004.

[75] M. Spreitzer and M. Theimer. Providing location information in a ubiquitous computing environment. In *Proceedings of 14th ACM Symposium on Operating System Principles (SIGOPS)*, volume 27, pages 270–283, 1993.

[76] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Procedings of First IEEE/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, pages 194–205, 2005.

[77] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald. Privacy aware location sensor networks. In *Proceedings of 9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX) ,Lihue, Hawaii*, volume 9, page 28, 2003.

[78] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, and S. Yi. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *Proceedings of 22nd IEEE International Conference of Distributed Computing Systems (ICDCS)*, pages 74–83, 2002.

[79] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *First IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, pages 113–124, 2005.

[80] J. Deng, R. Han, and S. Mishra. Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks. In *Elsevier Pervasive and Mobile Computing Journal, Special Issue on Security in Wireless Mobile Computing Systems*, volume 2, pages 159–186, 2006.

[81] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pages 113–126, Washington, DC, USA, 2005. IEEE Computer Society.

[82] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 599–608, Washington, DC, USA, 2005. IEEE Computer Society.

[83] C. Ozturk, Y. Zhang, and W. Trappe. Source-location privacy in energy-constrained sensor network routing. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (2004)*, pages 88–93, 2004.

[84] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao. Towards event source unobservability with minimum network traffic in sensor networks. In *Proceedings of the first ACM conference on Wireless network security (WISEC 2008), Alexandria, VA, USA*, pages 77–88, 2008.

[85] Y.X. Schwiebert and L.W. Shi. Preserving source location privacy in monitoring-based wireless sensor networks. In *Processing of 2nd International Workshop on Security in Systems and Networks (SSN), in conjnuction with Parallel and Distributed Processing Symposium (IPDPS)*, page 8, 2006.

[86] P. Kyasanur and N. Vaidya. Selfish mac layer misbehavior in wireless networks. In *IEEE Transactions on Mobile Computing*, volume 4, pages 502–516, 2005.

[87] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the confidant protocol: Cooperation of nodes - fairness in distributed ad-hoc networks. In *Proceedings of Mobile Internet Workshop. Informatik 2002. Dortmund, Germany*, pages 226–236, 2002.

[88] X. Huang, H. Zhai, and Y. Fang. Lightweight robust routing in mobile wireless sensor networks. In *Military Communications Conference (MILCOM 2006)*, pages 1–6, 2006.

[89] L. Buttyan and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of IEEE/ACM First Annual Workshop on Mobile and Ad Hoc Networking and Computing, (MobiHOC 2000), Boston, MA, USA*, pages 87–96, 2000.

[90] Levente Buttyan and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *MONET Journal of Mobile Networks*, 9(5):579–592, 2003.

[91] J. Deng, R. Han, and S. Mishra. Defending against path-based dos attacks in wireless sensor networks. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005), Alexandria, VA, USA*, pages 89–96, 2005.

[92] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehaviour in mobile ad hoc networks. In *Proceedings of the Conference on Mobile Computing and Networking*, pages 255–265, 2000.

[93] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *The First ACM Conference on Embedded Networked Sensor Systems (SenSys '03), Los Angeles, California, USA*, pages 255–265, 2003.

[94] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In *Proceedings of INFOCOM '04*, pages 2446–2457, 2004.

[95] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symposium on Security and Privacy (2004), Oakland, CA, USA*, pages 259–271, 2004.

[96] L. Lamport. Password authentication with insecure communication. In *Communications of the ACM*, number 11, pages 770–772, 1981.

[97] Y.-C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, pages 12–23, 2002.

[98] M. A. Hamid, M.-O. Rashid, and C.S. Hong. Routing security in sensor network: Hello flood attack and defense. In *IEEE ICNEWS 2006, Dhaka*, 2006.

[99] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *Wireless Networks*, 13(1):27–59, 2007.

[100] K. B. Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, 2007.

[101] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. Technical report, Proceedings of NDSS, 2004.

[102] S. Capkun, L. Buttyan, and J. Hubaux. Sector: Secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of Security of Ad Hoc and Sensor Networks (SASN '03)*, pages 21–32, 2003.

[103] L. Buttyan, L. Dora, and I. Vajda. Statistical wormhole detection in sensor networks. In *Security and Privacy in Ad-hoc and Sensor Networks*, volume 3813/2005, pages 128–141. Springer Berlin / Heidelberg, 2005.

[104] W. Wang, J. Kong, B. Bhargava, and M. Gerla. Visualization of wormholes in sensor networks. In *Proceedings of the 2004 ACM workshop on Wireless security (WISE)*, pages 51–60, 2004.

[105] R. Maheshwari, J. Gao, and S.R. Das. Detecting wormhole attacks in wireless networks using connectivity information. In *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pages 107–115, 2007.

[106] I. Khalil, S. Bagchi, and N. B. Shroff. LITEWORP: A lightweight countermeasure for the wormhole attack in multihop wireless networks. In *International Conference on Dependable Systems and Networks (DSN)*, pages 612–621, 2005.

[107] I. Khalil, S. Bagchi, and N.B. Shroff. Liteworp: Detection and isolation of the wormhole attack in static multihop wireless networks. In *Computer Networks*, number 13, pages 3750–3772, 2007.

[108] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H Rubens. Mitigating byzantine attacks in adhoc wireless networks. Technical Report Version 1, Department of Computer Science, Johns Hopkins University, 2004.

[109] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, volume 3, pages 1976–1986, 2003.

[110] W. Wang, B. Bhargava, Y. Lu, and X. Wu. Defending against wormhole attacks in mobile ad hoc networks. In *Wiley Journal on Wireless Communications and Mobile Computing*, volume 5, pages 1–21, 2005.

[111] F. Nait-Abdesselam, B. Bensaou, and T. Taleb. Detecting and avoiding wormhole attacks in wireless ad hoc networks. In *IEEE Communications Magazine*, volume 4, page 46, 2008.

[112] H. Alzaid, S. Abanmi, S. Kanhere, and C. T. Chou. Detecting wormhole attacks in wireless sensor networks. *Technical Report, Computer Science and Engineering School - UNSW, The Network Research Laboratory*, 2006.

[113] J. Eriksson, S.V. Krishnamurthy, and M. Faloutsos. Truelink: A practical countermeasure to the wormhole attack in wireless networks. In *Proceedings of the 14th IEEE International Conference on Network Protocols, 2006. ICNP '06.*, pages 75–84, 2006.

[114] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *ACM Workshop on Wireless Security (WISE03)*, pages 1–10, 2003.

[115] R. Shokri, M. Poturalski, G. Ravot, P. Papadimitratos, and J.-P. Hubaux. A practical secure neighbor verification protocol for wireless sensor networks. In *Proceedings of the Second ACM Conference on Wireless Network Security (WISEC '09), Zurich, Switzerland*, pages 193–200. ACM New York, NY, USA, 2009.

[116] R. Shokri, M. Poturalski, G. Ravot, P. Papadimitratos, and J.-P. Hubaux. A low-cost secure neighbor verification protocol for wireless sensor networks. Technical Report EPFL LCA-REPORT-2008-020, Laboratory for Computer Communications and Applications EPFL, Switzerland, 2008.

[117] E. C.-H. Ngai, J. Liu, , and M.R. Lyu. An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks. In *Computer Communications*, volume 30, pages 2353–2364, 2007.

[118] A.A. Pirzada and C.S. Mcdonald. Circumventing sinkholes and wormholes in ad-hoc wireless networks. In *Proceedings of International Workshop on Wireless Ad-hoc Networks, London, England, Kings College, London*, 2005.

[119] J. Douceur. The sybil attack. *Book Series Lecture Notes in Computer Science*, 2429/2002:251–260, 2002.

[120] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis defenses. In *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN '04), Berkeley, California, USA*, pages 259–268, 2004.

[121] L.A. Martucci, M. Kohlweiss, C. Andersson, and A. Panchenko. Self-certified sybil-free pseudonyms. In *Proceedings of the first ACM conference on Wireless network security (WISEC '08), Alexandria, VA, USA*, pages 154–159, 2008.

[122] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the IEEE Security and Privacy Symposium 2003*, pages 197–213. Berkeley, CA, United States, 2003. Institute of Electrical and Electronics Engineers Inc, 2003.

[123] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communication Security (CCS-02), Washington DC*, pages 41–47, 2002.

[124] Y.-C. Hu, A. Perrig, and D.B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WISE '03)*, pages 30–40, 2003.

[125] J. McCune, E. Shi, A. Perrig, and M. Reiter. Detection of denial-of-message attacks on sensor network broadcasts. In *IEEE Symposium on Security and Privacy*, pages 64–78, 2005.

[126] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe '02)*, 2002.

[127] R.L. Pickholtz, D.L. Schilling, and L.B. Milstein. Theory of spread spectrum communicationsa tutorial. In *IEEE Transactions on Communications*, number 5, pages 855–884, 1982.

[128] R. Negi and A. Perrig. Jamming analysis of mac protocols. *Carnegie Mellon Technical Memo*, 2003.

[129] R. Mallik, R. Scholtz, and G. Papavassilopoulos. Analysis of an on-off jamming situation as a dynamic game. In *IEEE Trans. Commun.*, number 8, pages 1360–1373, 2000.

[130] J. Jung, V. Paxson, A.W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proceedings IEEE Symposium on Security and Privacy*, pages 211–225, 2004.

[131] V. Coskun, E. Cayirci, A. Levi, and S. Sancak. Quarantine region scheme to mitigate spam attacks in wireless-sensor networks. In *IEEE Trans. on Mobile Computing*, volume 5, pages 1074–1086, 2006.

[132] Y. W. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy efficient link-layer jamming attacks against wireless sensor network mac protocols. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks (SASN), Alexandria, VA, USA*, pages 76–88, 2005.

[133] Y.W. Law, P.H. Hartel, J.I. den Hartog, and P.J.M. Havinga. Link-layer jamming attacks on s-mac. In *Proceedings of EWSN*, pages 217–225, 2005.

[134] T.X. Brown, J.E. James, and A. Sethi. Jamming and sensing of encrypted wireless ad hoc networks. In *Proceedings of the Seventh ACM international symposium on Mobile ad hoc networking and computing, Florence, Italy*, pages 120–130, 2006.

[135] A.D. Wood, J.A. Stankovic, and S.H. Son. Jam: A jammed-area mapping service for sensor networks. In *24th IEEE International Real-Time Systems Symposium (RTSS)*, pages 286–297, 2003.

[136] M. Cagalj, S. Capkun, and J.-P. Hubaux. Wormhole-based anti-jamming techniques in sensor networks. In *IEEE Trans. on Mobile Computing*, volume 6, pages 100–114, 2007.

[137] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings Workshop on Wireless Security (WiSe '04)*, pages 80–89, 2004.

[138] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of MobiHoc. ACM Press*, pages 46–57, 2005.

[139] E. Ayday, Farshid D., and F. Fekri. Location-aware security services for wireless sensor networks using network coding. In *26th IEEE International Conference on Computer Communications (INFOCOM '07), Anchorage, AK, USA*, pages 1226–1234. IEEE, 2007.

[140] G. Zhou, C. Huang, T. Yan, T. He, J.A. Stankovic, and T. F. Abdelzaher. Mmsn: Multi-frequency media access control for wireless sensor networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–13, 2006.

[141] A. Wood, J. Stankovic, and G. Zhou. Deejam: Defeating energy-efficient jamming in ieee 802.15.4-based wireless networks. In *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,(SECON '07)*, pages 60–69, 2007.

[142] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *26th IEEE International Conference on Computer Communications (INFOCOM '07),Anchorage, AK, USA*, pages 1307–1315, 2007.

[143] J. M. McCune, E. Shi, A. Perrig, and M. K. Reiter. Detection of denial of message attacks on sensor network broadcasts. In *Proceedings IEEE Symposium on Security and Privacy*, pages 64–78, 2005.

[144] G. Lin and G. Noubir. On link-layer denial of service in data wireless lans. In *Journal on Wireless Communications & Mobile Computing*, volume 5, pages 273–284, 2005.

[145] Z. Karakehayov. Using reward to detect team black-hole attacks in wireless sensor networks. In *Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN'05), Stockholm, Sweden*, pages 74–78, 2005.

[146] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *The First ACM Conference on Embedded Networked Sensor Systems (Sensys'03), Los Angeles, California, USA*, pages 14–27, 2003.

[147] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (Mobicom '2000), Boston, MA*, pages 255–265, 2000.

[148] A. A. Pirzada and C. McDonald. Establishing trust in pure ad-hoc networks. In *Proceedings of 27th Australasian Computer Science Conference (ACSC'04)*, pages 47–54, 2004.

[149] I. Khalil, S. Bagchi, and C. Nita-Rotaru. Dicas: Detection, diagnosis and isolation of control attacks in sensor networks. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, pages 89–100, 2005.

[150] J. Deng, R. Han, and S. Mishra. A robust and lightweight routing mechanism for wireless sensor networks. In *Wireless Ad Hoc Networks and Sensor Networks*, 2004.

[151] J. Yin and S. K. Madria. A hierarchical secure routing protocol against black hole attacks in sensor networks. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (2006).*, volume 1, pages 376 – 383, 2006.

[152] P. Papadimitratos and Z. J. Haas. Secure message transmission in mobile ad hoc networks. In *Adhoc Networks*, pages 193–209, 2003.

[153] J. Staddon, D. Balfanz, and G. Durfee. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA'02), Atlanta, Georgia, USA*, pages 122–130, 2002.

[154] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN 02), Fairfax, Virginia*, pages 135–147, 2002.

[155] O. Obst. Distributed fault detection using a recurrent neural network. In *International Conference on Information Processing in Sensor Networks (IPSN 2009), San Francisco, CA, USA*, pages 373–374, 2009.

[156] J. Staddon, D. Balfanz, and G. Durfee. Efficient tracing of failed nodes in sensor networks. In *ACM WSNA'02,Atlanta, GA*, pages 122–130, 2002.

[157] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy. Secure locations: Routing on trust and isolating compromised sensors in location-aware sensor networks. In *ACM SenSys'03, Los Angeles, California*, pages 324–325, 2003.

[158] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *Proceedings of 24th Annual Joint Conference of the IEEE*

Computer and Communications Societies (INFOCOM 2005), volume 2, pages 902–913, 2005.

[159] T. Clouqueur, K.K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. In *IEEE Transactions on Computers*, number 3, pages 320–333, 2004.

[160] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. In *IEEE Transactions on Computers*, number 3, pages 241–250, 2004.

[161] K.K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. In *IEEE Ad Hoc Networks Journal*, pages 59–70, 2003.

[162] Y. Zhao, R. Govindan, and D. Estrin. Residual energy scans for monitoring wireless sensor networks. In *IEEE WCNC'02, Florida*, pages 78–89, 2002.

[163] S.J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *IEEE International Conference on Communications (ICC 2001)*, volume 10, pages 3201–3205, 2001.

[164] D.P. Vincent and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM*, pages 1405–1413, 1997.

[165] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient energy-efficient multipath routing in wireless sensor networks. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 5, pages 11–25, 2001.

[166] S. De, C. Qiao, and H. Wu. Meshed multipath routing: An efficient strategy in sensor networks. In *Proceedings of the Wireless Communications and Networking Conference, IEEE (WCNC'03)*, volume 3, pages 481–497, 2003.

[167] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[168] Nasipuri and S. Das. On-demand multipath routing for mobile ad-hoc networks. In *8th International Conference on Computer Communications and Networks (ICCN 99)*, pages 64–70, 1999.

[169] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *IEEE Proceedings of the Wireless Communications and Networking Conference (WCNC'03), New Orleans, LA, USA*, volume 3, pages 1918–1922, 2003.

[170] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad-hoc routing for wireless networks. In *Proceedings of the ACM international Conference on Mobile ad hoc networking and computing (MobiHoc '01)*, pages 299–302, 2001.

[171] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04). NewYork, NY, USA: ACM Press*, pages 162–175, 2004.

[172] ZigBee Alliance. Zigbee specification. *Technical Report Document 053474r06, Version 1.0, ZigBee Alliance*, 2005.

[173] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: A secure sensor network communication architecture. In *6th International Symposium on Information Processing in Sensor Networks,(IPSN '07) Cambridge, MA,USA*, pages 479–488, 2007.

[174] R.D. Pietro, L.V. Mancini, Y.W. Law, S. Etalle, and P. Havinga. LKHW: A directed diffusion-based secure multicast scheme for wireless sensor networks. In *2003 International Conference on Parallel Processing Workshops (ICPPW'03)*, pages 397–406, 2003.

[175] P. Traynor, R. Kumar, H.B. Saad, G. Cao, and T.L. Porta. Liger: Implementing efficient hybrid security mechanisms for heterogeneous sensor networks. In *Proceedings of the 4th international conference on Mobile systems, applications and services (MOBISYS '06), Uppsala, Sweden*, pages 15–27, 2006.

[176] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La-Porta. Establishing pair-wise keys in heterogeneous sensor networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006). Barcelona, Spain*, pages 1–12, 2006.

[177] P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu, and T. La Porta. Efficient hybrid security mechanisms for heterogeneous sensor networks. In *IEEE Transactions on Mobile Computing*, volume 6, pages 663–677, 2007.

[178] P.E. Lanigan, R. Gandhi, and P. Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006)*, pages 53–62, 2006.

[179] J. Huang, J. Buckingham, and R. Han. Level key infrastructure for secure and efficient group communication in wireless sensor networks. In *First IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, pages 249–260, 2005.

[180] N. Gui, R. Chen, Z. Cai, J. Hu, and Z. Cheng. A secure routing and aggregation protocol with low energy cost for sensor networks. In *International Symposium on Information Engineering and Electronic Commerce, 2009 (IEEC '09)*, pages 79–84, 2009.

[181] Y. Yang, X. Wang, S. Zhu, and G. Cao. Sdap: a secure hop-by-hop data aggregation protocol for sensor networks. In *Proceedings of the Seventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc '06), Florence, Italy*, pages 356–367, 2006.

[182] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: Securing sensor networks with public key technology. *Workshop on Security of Ad Hoc and Sensor Networks*, pages 59–64, 2004.

[183] G. Gaubatz, J.-P. Kaps, E. Ozturk, and B. Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *2nd IEEE International Workshop on Pervasive Computing and Communication Security, Kauai Island, Hawaii, USA*, 2005.

[184] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet. In *Third Annual IEEE International Conference on Pervarsive Computing and Communications (PerCom), Kauai Island, Hawaii,USA*, pages 247–256, 2005.

[185] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *International Workshop on Cryptographic Hardware and Embedded Systems*, volume 31, pages 119–132, 2004.

[186] D.J. Malan, M. Welsh, and M.D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, pages 71–81, 2004.

[187] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *In International Conference on Information Processing in Sensor Networks (IPSN '08)*, pages 245–256, 2008.

[188] G. Bertoni, L. Breveglieri, and M. Venturi. ECC Hardware Coprocessors for 8-bit Systems and Power Consumption Considerations. In *Third International Conference on Information Technology: New Generations (ITNG 2006)*, pages 573–574, 2006.

[189] B. Doyle, S. Bell, A. F. Smeaton, K. McCusker, and N. E. OConnor. Security considerations and key negotiation techniques for power constrained sensor networks. *The Computer Journal*, 49:443–453, 2006.

[190] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *Wireless sensor networks*, volume LNCS 4913, pages 305–320, 2008.

[191] L.B. Oliveira, M. Scott, J. Lopez, and R. Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *5th International Conference on Networked Sensing Systems (INSS 2008)*, pages 173–180, 2008.

[192] Q. Dong, D. Liu, and P. Ning. Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks. In *Proceedings of the first ACM conference on Wireless network security (WISEC '08), Alexandria, VA, USA*, pages 2–12, 2008.

[193] B. Driessen, A. Poschmann, and C. Paar. Comparison of innovative signature algorithms for wsns. In *Proceedings of the first ACM conference on Wireless network security (WISEC '08), Alexandria, VA, USA*, pages 30–35, 2008.

[194] L. Yuan and G. Qu. Design space exploration for energy-efficient secure sensor network. In *The IEEE International Conference on Application-Specific Systems Architectures and Processors*, pages 88–97, 2002.

[195] S. Chang, S. Shieh, W.W. Lin, and C. Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *Proceedings of the 2006 ACM Symposium on information (ASIACCS '06), Computer and Communications Security (Taipei, Taiwan)*, pages 311–320. CM Press, New York, NY, 2006.

[196] G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2004.

[197] G. Behrmann, A. David, K.G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST) 2006*, IEEE Computer Society, pages 125–126, 2006.

[198] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In *Proceedings PAPM/PROBMIV'01 Tools Session*, pages 7–12, 2001.

[199] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004.

[200] M. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *Proceedings 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pages 322–323. IEEE Computer Society Press, 2004.

[201] M. Naughton, D. Heffernan, and G. Leen. Use of timed automata models in the design of real-time control network elements. In *ETFA '06*, pages 433–436, 2006.

[202] G. Leen and D. Heffernan. Modelling and formal verification of a time-triggered network protocol. In *International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICN/ICONS/MCL 2006)*, pages 178–189, 2006.

[203] A. Fehnker, M. Fruth, and A. McIver. Graphical modeling for simulation and formal analysis of wireless network protocols. *Technical Report : MeMot*, 2007.

[204] L. Samper, F. Maraninchi, L. Mounier, and L. Mandel. Glonemo: global and accurate formal models for the analysis of ad-hoc sensor networks. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks table of contents, Nice, France*, number 3, pages 138–145, 2006.

[205] T.F. Smit. Verification of sensor network models using uppaal. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2005. Supervised by Prof. Jan Madsen.

[206] J.A. Stine and G. de Veciana. A paradigm for quality-of-service in wireless ad hoc networks using synchronous signaling and node states. *IEEE Journal on Selected Areas in Communications*, 22:1301 – 1321, 2004.

[207] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala, editors, *Proceedings 2nd Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'02)*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

[208] O. Sharma, J. Lewis, A. Miller, A. Dearle, D. Balasubramaniam, R. Morrison, and J. Sventek. Towards verifying correctness of wireless sensor network applications using insense and spin. In *16th International SPIN Workshop on Model Checking of Software (SPIN 2009), Grenoble, France*, pages 223–240. Springer 2009, 2009.

[209] M.S. Silva, F. Martins, L. Lopes, and J. Barros. A calculus for sensor networks. *Technical Report : CoRR abs/cs/0612093*, 2006.

[210] F. Heidarian, J. Schmaltz, and F.W. Vaandrager. Analysis of a clock synchronization protocol for wireless sensor networks. In *16th International Symposium of Formal Methods (FM2009), Eindhoven, the Netherlands*. Lecture Notes in Computer Science. Springer, 2009.

[211] M. Schuts, F. Zhu, F. Heidarian, and F.W. Vaandrager. Modelling clock synchronization in the chess gmac wsn protocol. *QFM, Eindhoven*, 2009.

[212] D. Camara, A.A.F. Loureiro, and F. Filali. Methodology for formal verification of routing protocols for ad hoc wireless networks. In *IEEE Global Telecommunications Conference, 2007 (GLOBECOM '07)*, pages 705–709, 2007.

[213] A. Fehnker, L. van Hoesel, and A. Mader. Modelling and verification of the lmac protocol for wireless sensor networks. In *IFM*, pages 253–272, 2007.

[214] W. Henderson and S. Tron. Verification of the minimum cost forwarding protocol for wireless sensor networks. In *11th IEEE International Conference Emerging Technologies and Factory Automation, Prague*, 2006.

[215] R. Cardell-Oliver. Why flooding is unreliable (extended version). *Technical Report UWA-CSSE-04-001, CSSE, University of Western Australia*, 2001.

[216] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *Principles of Mobile Computing 2002, Toulouse, France*, 2002.

[217] P. Downey and R. Cardell-Oliver. Evaluating the impact of limited resource on the performance of flooding in wireless sensor networks. In *International Conference on Dependable Systems and Networks (2004)*, pages 785 – 794, 2004.

[218] J. Heidemann, F. Silva, and D. Estrin. Matching data dissemination algorithms to application requirements. In *Proceedings of the first international conference on Embedded Networked Sensor Systems (ACM Press)*, pages 218–229, 2003.

[219] B. Krishnamachari and J. Heidemann. Application-specific modelling of information routing in wireless sensor networks. In *USC ISI Technical report isi-tr-676*, 2003.

[220] S. Nair and R. Cardell-Oliver. Formal specification and analysis of performance variation in sensor network diffusion protocols. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 170–173, 2004.

[221] L. Lamport. The temporal logic of actions. In *ACM Transactions on Programming Languages and Systems*, number 3, pages 872–923, 1994.

[222] M. Abadi and L. Lamport. Conjoining specifications. In *ACM Transactions on Programming Languages and Systems*, number 3, pages 507–534, 1995.

[223] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, USA*, pages 27–31, 2002.

[224] M. Burmester and B. de Medeiros. Towards provable security for route discovery protocols in mobile ad hoc networks. In *ACM Transacions on Mobile Computing*, 2009.

[225] Y.-C. Hu, A. Perrig, and D. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of Network and Distributed System Security Symp. (NDSS'03)*, pages 57–73, 2003.

[226] N. Asokan M.G. Zapata. Securing ad hoc routing protocols. In *Proc of WiSe'02, Atlanta, Georgia, USA*, pages 1–10, 2002.

[227] T.R. Andel and A. Yasinsac. Automated security analysis of ad hoc routing protocols. In *Proceedings of the Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'07), Wroclaw, Poland*, pages 9–26, 2007.

[228] T.R. Andel and A. Yasinsac. Adaptive threat modeling for secure ad hoc routing protocols. In *Proceedings of 3rd International Workshop on Security and Trust Management (STM-07), Dresden, Germany*, volume 197, pages 3–14, 2008.

[229] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *First ACM Conference on Embedded Networked Sensor Systems (Sensys03)*, pages 126–137, 2003.

[230] D. Gay, M. Welsh, P. Levis, E. Brewer, R. von Behren, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of Programming Language Design and Implementation (PLDI)*, pages 1–11, 2003.

[231] R. Baldwin and R. Rivest. The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS algorithms. Technical report, Department of Computer Science, Rice University, 1996.

[232] Y.W. Law, S. Dulman, S. Etalle, and P. Havinga. Assessing security-critical energy-efficient sensor networks. CTIT technical reports series 02-18 (TR-CTIT), 2003.

[233] J.L. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, and K.S.J. Pister. System architecture directions for networked sensors. In *Proceedings of Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, pages 93–104, 2000.

[234] A.-S.K. Pathan, H.-W. Lee, and C.S. Hong. Security in wireless sensor networks: Issues and challenges. In *Proceedings of 8th IEEE ICACT 2006, Phoenix Park, Korea*, volume 2, pages 1043–1048, 2006.

[235] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Raeed- a formally evaluated routing protocol for wsn against dos attacks. 2010.

[236] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Formal specifications of denial of service attacks in wireless sensor networks. Submitted at Pervasive and Mobile Computing Journal, 2010.

[237] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. An innovative solution for the ina and wormhole attack in wireless sensor networks (wsns). 2010.

[238] K. Saghar, W. Henderson, and D. Kendall. Formal modelling and analysis of routing protocol security in wireless sensor networks. In *PGNET '09*, pages 73–78, 2009.

[239] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Formal modelling of a robust wireless sensor network routing protocol. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2010)*, 2010.

[240] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Applying formal modelling to detect dos attacks in wireless medium. In *IEEE, IET International Symposium on COMMUNICATION SYSTEMS, NETWORKS AND DIGITAL SIGNAL PROCESSING NASA/ESA(CSNDSP 2010)*, 2010.

[241] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Vulnerability of insens to denial of service attacks. 2010.

[242] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Automatic detection of black hole attack in wireless network routing protocols. 2010.

[243] D. Dolev and A. Yao. On the security of public key protocols. In *IEEE Transactions on Information Theory*, volume 29, pages 198–208, 1983.

[244] K. Fall and K. Varadhan. Ns-2: Network simulator-2. Technical report, UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2007.

[245] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: a library for parallel simulation of large-scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations - PADS '98*, pages 154–161, 1998.

[246] Andras Varga. Omnet++: Objective modular network testbed in c++. 2003.

[247] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J.S. Baras. Atemu: A fine-grained sensor network simulator. In *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, (SECON '04)*, pages 145–152, 2004.

[248] B.L. Titzer, D.K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN '05)*, pages 477–482, Piscataway, NJ, USA, 2005. IEEE Press.

[249] ILLINOIS. J-sim. The OHIO State University, 2007.

[250] Visaul Sense. University of California at Berkeley, 12 Febraury 2007.

[251] Sameer Sundresh, Wooyoung Kim, and Gul Agha. Sens: A sensor, environment and network simulator. In *37th Annual Simulation Symposium (ANSS37)*, 2004.

[252] Alexander Kroeller, Dennis Pfisterer, Carsten Buschmann, Sandor P. Fekete, and Stefan Fischer. Shawn: A new approach to simulating wireless sensor networks. Technical report, 2005. "http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0502003".

[253] L.F. Perrone and D. Nicol. A scalable simulator for tinyos applications. Proceedings of the 2002 Winter Simulation Conference, 2002.

[254] Gyula Simon. Prowler : Probabilistic wireless network simulator. Vanderbilt University, 2004.

[255] Akos Ledeczi. Jprowler. DARPA, 2005.

[256] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 21–30. ACM Press, 2007.